NS-UDPD: A FINITE-DOMAIN JOINT SELECTION METHOD FOR LAYER-WISE SPARSITY ALLOCATION IN LLM PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Layer-wise sparsity allocation improves unstructured post-training pruning of large language models (LLMs) by distributing a global sparsity budget across Transformer layers. Existing metric-based methods overlook pruning-cost variations across sparsity levels. Search-based and learning-based methods lack global optimality guarantees for discrete allocations, while reconstruction-based methods optimize Euclidean error rather than the language-modeling negative log-likelihood (NLL) objective. We therefore propose NS-UDPD, a finite-domain joint selection method that selects one candidate sparsity level per layer while exactly matching the target mean layer sparsity. Diagonal-Kronecker NLL Sensitivity (DKNS) combines squared weights, input-activation second moments, and squared output-NLL-gradient energies into per-weight DKNS proxy deletion costs, which candidate deletion masks aggregate into layer-level DKNS state costs. Cost-Only Allocation may concentrate state costs in shallow layers, where pruning perturbations propagate through subsequent layers. Uniform Depth-Prefix Dominance (UDPD) therefore bounds the cumulative state cost at every depth prefix using the uniform reference allocation. A three-level lexicographic objective first minimizes the full-depth DKNS state cost and then improves the depth-prefix cost distribution and layer-wise sparsity balance. We evaluate NS-UDPD on five LLaMA-3 and Qwen3.5 series models under three fixed local pruning rules and 40%–70% global unstructured sparsity. At 70% sparsity, it reduces WikiText-2 perplexity relative to the best competing allocation on Qwen3.5-2B with SparseGPT by 31.55% and raises six-task mean accuracy from 37.20% to 41.26% on Qwen3.5-4B with Wanda relative to Uniform.

1 INTRODUCTION

Large language models (LLMs) deliver strong performance across language understanding, generation, and reasoning tasks (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023; OpenAI, 2023). However, their scale incurs substantial memory, storage, and inference costs, hindering efficient deployment. Quantization and pruning have been widely studied to reduce these costs (Frantar et al., 2022; Yao et al., 2022; Xiao et al., 2023; Frantar & Alistarh, 2023; Sun et al., 2024). This work focuses on unstructured post-training pruning, which sparsifies pretrained LLMs using a small calibration set without full retraining. Complementing the within-module weight selection of SparseGPT and Wanda (Frantar & Alistarh, 2023; Sun et al., 2024), layer-wise sparsity allocation distributes the global sparsity budget across Transformer layers according to their pruning sensitivities (Yin et al., 2024; Chen et al., 2025; Lu et al., 2024; Li et al., 2024b).

Existing layer-wise sparsity allocation methods follow three overlapping paradigms. Metric-based methods derive non-uniform sparsity rates from outlier, weight–activation, spectral, or activation-correlation statistics, as in OWL, DLP, AlphaPruning, and ALS (Yin et al., 2024; Chen et al., 2025; Lu et al., 2024; Li et al., 2024b). Search-based and learning-based methods, including DSA, BESA, and Lua-LLM, discover allocation functions or optimize differentiable sparsity variables (Li et al., 2024a; Xu et al., 2024; Lu et al., 2025). Reconstruction-based methods, including LSA and ATP, guide allocation using local linear reconstruction errors or depth-wise error propagation (Yang et al.,

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.

2026; Huang et al., 2025). Collectively, these studies show that adapting the global sparsity budget to heterogeneous pruning sensitivities across layers is critical for preserving model performance.

However, these paradigms retain distinct limitations. Metric-based methods overlook variations in pruning cost across candidate sparsity levels. Search- and learning-based methods rely on heuristic search or differentiable relaxations without formal global-optimality guarantees for the resulting discrete allocation. Reconstruction-based methods model local responses or propagated errors but optimize Euclidean reconstruction error rather than language-modeling NLL objective. These limitations motivate us to formulate a finite-domain joint selection problem, which selects one candidate sparsity level per layer to minimize the full-depth NLL-sensitive state cost, while exactly matching the target mean layer sparsity and bounding cumulative state cost at every depth prefix.

To formalize this problem, we discretize the sparsity range into candidate sparsity levels, each defining a candidate layer–sparsity state for every layer. Under the fixed local pruning rule, each state has an NLL-sensitive layer-level state cost. We select exactly one state per layer to minimize the full-depth state cost while exactly matching the target mean layer sparsity. Uniform prefix bounds further constrain cumulative state cost of the selected allocation at every depth prefix. The final joint selection problem is a multiple-choice 0–1 integer linear program over the discrete candidate set.

Building on this formulation, we propose NS-UDPD, a finite-domain joint selection method with Diagonal-Kronecker NLL Sensitivity (DKNS) and Uniform Depth-Prefix Dominance (UDPD) for layer-wise sparsity allocation. Existing methods typically quantify layer-wise pruning perturbations using Euclidean error. However, this surrogate does not directly reflect the language-modeling NLL objective that governs next-token prediction. To address this mismatch, DKNS combines squared weights, input-activation second moments, and squared output-NLL-gradient energies into per-weight DKNS proxy deletion costs. Candidate deletion masks induced by the fixed local pruning rule select and aggregate these costs into layer-level DKNS state costs, which better capture the effect of pruning on the language-modeling quality of LLMs. However, Cost-Only Allocation may concentrate state costs in shallow layers, whose pruning perturbations propagate through subsequent layers. UDPD controls the accumulation of state costs with depth through prefix-wise comparison with the uniform reference allocation. Finally, lexicographic joint selection first minimizes the full-depth DKNS state cost and then improves the depth-prefix cost distribution and layer-wise sparsity balance, yielding the final layer-wise sparsity allocation.

We evaluate NS-UDPD on five models from the LLaMA-3 and Qwen3.5 series under 40%–70% global unstructured sparsity. NS-UDPD consistently outperforms existing layer-wise sparsity allocation methods across different pruning rules, with larger gains at high sparsity ratios. At 70% sparsity, it reduces the best competing PPL on Qwen3.5-2B with SparseGPT by 31.55%. Relative to Uniform, NS-UDPD raises the six-task Mean from 37.20% to 41.26% on Qwen3.5-4B with Wanda. These results demonstrate that NS-UDPD preserves LLM performance under high sparsity.

2 METHOD

2.1 PROBLEM FORMULATION

Let θ^{dense} denote the parameters of a pretrained dense language model with L Transformer layers ordered according to the forward pass. Given a calibration set $\mathcal{D}$, a fixed local pruning rule P, and a target mean layer sparsity S, our goal is to jointly select the sparsity of each layer from a discrete candidate set while controlling the accumulation of state costs over network depth. Given a sparsity-grid resolution $Q \in \mathbb{N}_+$ and a sparsity range $[s_{\min}, s_{\max}]$, we define

$$\mathcal{T} = \{\lceil Qs_{\min} \rceil, \ldots, \lfloor Qs_{\max} \rfloor\}, \qquad s_t = \frac{t}{Q}, \quad t \in \mathcal{T}, \qquad t^\star = QS \in \mathcal{T}. \tag{1}$$

Here, $\mathcal{T}$ is the candidate state index set, s_t is a candidate sparsity level, and $t^\star$ is the uniform reference state corresponding to S. For layer ℓ, the candidate sparsity level s_t defines a candidate layer–sparsity state (ℓ, t). Under rule P, this state determines a candidate deletion set for each prunable Linear module in the layer. The NLL-sensitive costs of these candidate deletion sets are aggregated across Linear modules. Layer-wise rebasing then yields the rebased layer-level state cost $c_{\ell,t}^{(P)}$ used in the final joint selection problem. Let $x_{\ell,t} \in \{0, 1\}$ indicate whether layer ℓ selects state

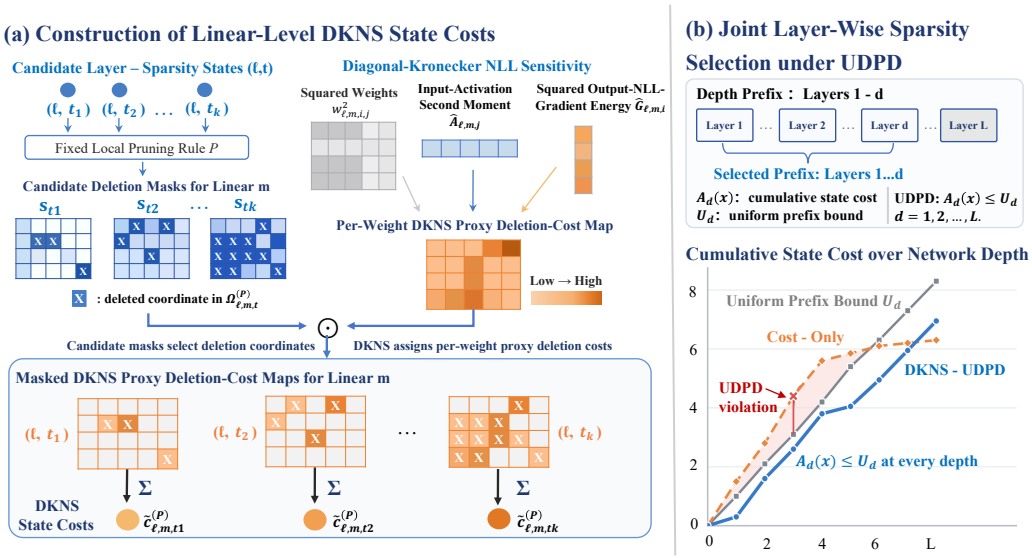


Figure 1: Overview of NS-UDPD. (a) DKNS combines squared weights, input-activation second moments, and squared output-NLL-gradient energies into per-weight proxy deletion costs, which candidate deletion masks select and aggregate into linear-level state costs. (b) UDPD jointly selects layer-wise sparsities while bounding every cumulative depth-prefix cost by the uniform prefix bound, preventing the shallow-layer cost concentration permitted by Cost-Only Allocation.

t, and let x collect all state-selection variables. The basic feasible set satisfying the one-state-per-layer constraint and the exact mean layer-sparsity constraint is

$$\mathcal{X}(S) = \left\{ x \;\middle|\; \begin{array}{ll} \displaystyle\sum_{t\in\mathcal{T}} x_{\ell,t} = 1, & \ell = 1,\ldots,L, \\ \displaystyle\sum_{\ell=1}^{L}\sum_{t\in\mathcal{T}} t\, x_{\ell,t} = Lt^\star, & \\ x_{\ell,t} \in \{0,1\}, & \ell = 1,\ldots,L, \quad t \in \mathcal{T} \end{array} \right\}. \tag{2}$$

The first condition requires each layer to select exactly one candidate state. Since $s_t = t/Q$ and $t^\star = QS$, the second condition ensures that the mean of the selected layer-wise sparsities is exactly S. For the depth prefix ending at layer d, we define its cumulative state cost as

$$A_d(x) = \sum_{\ell=1}^{d}\sum_{t\in\mathcal{T}} c_{\ell,t}^{(P)} x_{\ell,t}, \qquad d = 1,\ldots,L. \tag{3}$$

This quantity aggregates the costs of the candidate states selected by the first d Transformer layers, while $A_L(x)$ is the total state cost over all layers. Given a uniform prefix bound U_d, the primary objective of the finite-domain joint layer-wise sparsity selection problem is to minimize the full-depth state cost:

$$\begin{aligned} \min_{x\in\mathcal{X}(S)} \quad & A_L(x) \\ \text{s.t.} \quad & A_d(x) \le U_d, \qquad d = 1,\ldots,L. \end{aligned} \tag{4}$$

Eq. 2 specifies the basic feasible set for candidate-state selection, whereas Eq. 4 further constrains the cumulative state cost of the selected allocation at every depth prefix. For fixed $c_{\ell,t}^{(P)}$ and U_d, this problem is a multiple-choice 0–1 integer linear program defined over the discrete candidate sparsity set. Section 2.2 constructs the raw layer-level DKNS state cost $\widetilde{c}_{\ell,t}^{(P)}$. Section 2.3 constructs U_d using the uniform reference allocation and Uniform Depth-Prefix Dominance (UDPD), and then performs lexicographic joint selection with Eq. 4 as its first-level objective.

2.2 Candidate-State Cost Modeling with Diagonal-Kronecker NLL Sensitivity

Section 2.1 restricts the sparsity of each layer to the discrete candidate set $\{s_t : t \in \mathcal{T}\}$. For a candidate layer–sparsity state (ℓ, t), the fixed local pruning rule P determines a candidate deletion

set $\Omega^{(P)}_{\ell,m,t}$ for each prunable Linear module m in layer ℓ. Its binary representation is the corresponding candidate deletion mask. Joint layer-wise sparsity selection requires comparable pruning costs across candidate states. However, local Euclidean reconstruction error applies the same metric to all output coordinates and does not account for differences in NLL sensitivity across output channels. Motivated by this limitation and inspired by Kronecker-factored approximate curvature (K-FAC) (Martens & Grosse, 2015), we use Diagonal-Kronecker NLL Sensitivity (DKNS) to construct a proxy state cost related to the language-modeling objective for each candidate deletion set. Fig. 1(a) summarizes how DKNS constructs linear-level state costs from the discrete candidate set for a single prunable Linear module.

Consider the m-th prunable Linear module in layer ℓ, with weight matrix $W_{\ell,m} \in \mathbb{R}^{d^{\text{out}}_{\ell,m} \times d^{\text{in}}_{\ell,m}}$ and output $y_{\ell,m,n,q} = W_{\ell,m} x_{\ell,m,n,q}$. Let $\mathcal{I}_{\ell,m,n}$ denote the set of valid positions for this module in the n-th calibration sequence z_n, and define $\nu_{\ell,m} = \sum_{n=1}^{N} |\mathcal{I}_{\ell,m,n}|$. We estimate the input-activation second moment and the squared output-NLL-gradient energy as

$$\begin{aligned} \widehat{A}_{\ell,m,j} &= \frac{1}{\nu_{\ell,m}} \sum_{n=1}^{N} \sum_{q \in \mathcal{I}_{\ell,m,n}} x^2_{\ell,m,n,q,j}, \\ \widehat{G}_{\ell,m,i} &= \frac{1}{N} \sum_{n=1}^{N} \sum_{q \in \mathcal{I}_{\ell,m,n}} \left(\frac{\partial \mathcal{L}_n}{\partial y_{\ell,m,n,q,i}} \right)^2 . \end{aligned} \tag{5}$$

Here, $\mathcal{L}_n$ is the token-averaged causal NLL of sequence z_n. The statistic $\widehat{A}_{\ell,m,j}$ measures the activation strength of input channel j along the dense-model calibration trajectory, whereas $\widehat{G}_{\ell,m,i}$ measures the local NLL sensitivity of output channel i. Let $D^A_{\ell,m} = \mathrm{Diag}(\widehat{A}_{\ell,m})$ and $D^G_{\ell,m} = \mathrm{Diag}(\widehat{G}_{\ell,m})$. We define the diagonal-Kronecker sensitivity matrix as

$$M^{\text{DKNS}}_{\ell,m} = D^A_{\ell,m} \otimes D^G_{\ell,m} \succeq 0. \tag{6}$$

Since both factors have nonnegative diagonal entries, $M^{\text{DKNS}}_{\ell,m}$ is positive semidefinite and induces a positive-semidefinite quadratic form over parameter perturbations. In a second-order Taylor expansion of the causal NLL around the current weight matrix $W_{\ell,m}$, the second-order term is governed by local curvature. We use $M^{\text{DKNS}}_{\ell,m}$ as a tractable positive-semidefinite curvature estimate. The second-order NLL term associated with a weight perturbation $\Delta W_{\ell,m}$ is then approximated by

$$\mathcal{S}^{\text{DKNS}}_{\ell,m}(\Delta W_{\ell,m}) = \frac{1}{2} \sum_{i=1}^{d^{\text{out}}_{\ell,m}} \sum_{j=1}^{d^{\text{in}}_{\ell,m}} \widehat{G}_{\ell,m,i} \widehat{A}_{\ell,m,j} \left(\Delta w_{\ell,m,i,j} \right)^2 . \tag{7}$$

When $w_{\ell,m,i,j}$ is set to zero, the corresponding perturbation is $\Delta w_{\ell,m,i,j} = -w_{\ell,m,i,j}$. Its per-weight DKNS proxy deletion cost is

$$d^{\text{DKNS}}_{\ell,m,i,j} = \frac{1}{2} w^2_{\ell,m,i,j} \widehat{G}_{\ell,m,i} \widehat{A}_{\ell,m,j}. \tag{8}$$

Eq. 8 jointly accounts for the magnitude of the deleted weight, the activation strength of its input channel, and the NLL sensitivity of its output channel. Thus, two connections with similar weight magnitudes and input-activation strengths can receive different costs when their output channels have different NLL sensitivities. DKNS is constructed from empirical-Fisher-style second-order statistics and provides a positive-semidefinite approximation to local NLL curvature. Appendix A.1 derives the diagonal-Kronecker approximation. Appendix A.2 further discusses its relationships to Taylor curvature, the empirical Fisher, Euclidean reconstruction error, and additivity bounds.

For a fixed Linear module, Eq. 8 defines a per-weight DKNS proxy deletion-cost map over all weight coordinates. This map is shared by all candidate sparsity levels of the module. For state (ℓ, t), the candidate deletion mask produced by the local pruning rule P selects the coordinates specified by $\Omega^{(P)}_{\ell,m,t}$ and forms a masked DKNS proxy deletion-cost map. The linear-level and raw layer-level DKNS state costs are defined as

$$\widetilde{c}^{(P)}_{\ell,m,t} = \sum_{(i,j) \in \Omega^{(P)}_{\ell,m,t}} d^{\text{DKNS}}_{\ell,m,i,j}, \qquad \widetilde{c}^{(P)}_{\ell,t} = \sum_{m \in \mathcal{M}_\ell} \widetilde{c}^{(P)}_{\ell,m,t}. \tag{9}$$

Here, $\mathcal{M}_\ell$ denotes the set of prunable Linear modules in layer ℓ. Enumerating all candidate states $t \in \mathcal{T}$ for a fixed layer ℓ yields its discrete sparsity–cost curve

$$\widetilde{\mathcal{C}}_\ell^{(P)} = \left\{ \left(s_t, \widetilde{c}_{\ell,t}^{(P)} \right) : t \in \mathcal{T} \right\}. \tag{10}$$

This curve describes how the DKNS state cost of layer ℓ changes with the candidate sparsity level under the fixed local pruning rule. Section 2.3 further applies layer-wise rebasing to $\widetilde{c}_{\ell,t}^{(P)}$ and uses the resulting costs to construct the Uniform Depth-Prefix Dominance (UDPD) constraints.

2.3 Joint Layer-Wise Sparsity Selection with Uniform Depth-Prefix Dominance

Section 2.2 constructs a raw layer-level DKNS state cost $\widetilde{c}_{\ell,t}^{(P)}$ for each candidate layer–sparsity state. Minimizing only the total state cost over all Transformer layers does not distinguish where the cost is incurred along network depth. It may therefore offset higher costs in shallow layers with lower costs in deeper layers. Since pruning perturbations introduced in shallow layers propagate through subsequent layers, such an allocation increases the risk of representation shifts throughout the network. To control the accumulation of state costs with depth, we introduce Uniform Depth-Prefix Dominance (UDPD). We apply layer-wise rebasing to the layer-level DKNS state costs:

$$c_{\ell,t}^{(P)} = \widetilde{c}_{\ell,t}^{(P)} - \min_{u \in \mathcal{T}} \widetilde{c}_{\ell,u}^{(P)}. \tag{11}$$

This transformation shifts the minimum candidate-state cost in each layer to zero. Because exactly one state is selected per layer, layer-wise rebasing preserves the full-depth cost ordering among allocations. It also preserves the unnormalized prefix-cost difference between each allocation and the uniform reference allocation. Appendix B.1 provides the proof. Under the grid assumption $s_t = t/Q$ from Section 2.1, the target mean layer sparsity S corresponds to the uniform reference state $t^\star = QS \in \mathcal{T}$. The uniform reference allocation selects $t^\star$ in every layer and therefore satisfies the same exact mean layer-sparsity constraint. For the depth prefix ending at layer d, we define the cumulative state cost of the current allocation and the uniform prefix bound as

$$A_d(x) = \sum_{\ell=1}^{d} \sum_{t \in \mathcal{T}} c_{\ell,t}^{(P)} x_{\ell,t}, \qquad U_d = \sum_{\ell=1}^{d} c_{\ell,t^\star}^{(P)}. \tag{12}$$

UDPD uses the cumulative state cost of the uniform reference allocation over the depth prefix ending at layer d as the uniform prefix bound U_d. It requires the current allocation to satisfy

$$A_d(x) \le U_d, \qquad d = 1, \dots, L. \tag{13}$$

Thus, the cumulative state cost of the current allocation cannot exceed that of the uniform reference allocation at any depth prefix. Fig. 1(b) illustrates how UDPD constrains the depth-wise distribution of state costs. Cost-Only Allocation minimizes only $A_L(x)$ and may therefore concentrate higher costs in shallow layers, yielding $A_d(x) > U_d$ at some depth prefixes. Under UDPD, any allocation that exceeds the uniform prefix bound at any depth prefix is excluded, thereby controlling the accumulation of state costs over network depth. This constraint neither requires layer-wise sparsity to vary monotonically with depth nor requires the state cost of every layer to be lower than that of the uniform reference allocation. The uniform reference allocation makes every UDPD constraint tight, which guarantees a nonempty feasible set.

Let $\mathcal{X}_{\mathrm{UDPD}}$ denote the feasible set satisfying the one-state-per-layer constraint, the exact mean layer-sparsity constraint, the binary selection constraints, and all UDPD constraints. Define the positive-bound prefix index set as $\mathcal{D}_+ = \{d \in \{1, \dots, L\} \mid U_d > 0\}$. When $\mathcal{D}_+ \neq$, define the worst normalized depth-prefix ratio as $R(x) = \max_{d \in \mathcal{D}_+} A_d(x)/U_d$; otherwise, set $R(x) = 0$. NS-UDPD adopts the following three-level lexicographic objective:

$$\operatorname*{lexmin}_{x \in \mathcal{X}_{\mathrm{UDPD}}} \left(A_L(x),\ R(x),\ \sum_{\ell=1}^{L} \sum_{t \in \mathcal{T}} (t - t^\star)^2 x_{\ell,t} \right). \tag{14}$$

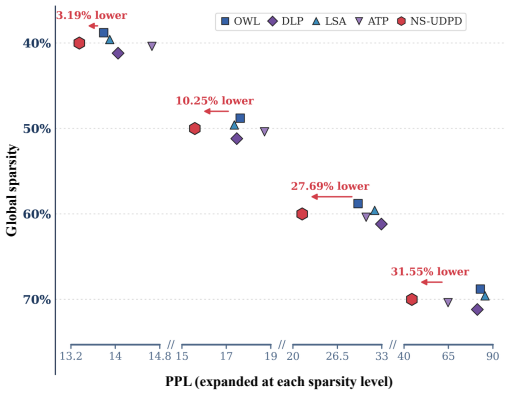


Figure 2: WikiText-2 PPL on Qwen3.5-2B with SparseGPT across four global sparsity levels. Lower values indicate better overall language-modeling performance after pruning.

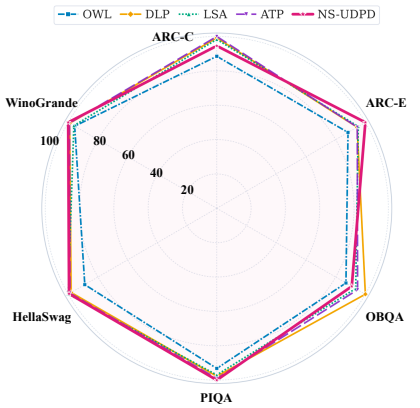


Figure 3: Normalized task-wise zero-shot accuracy on six tasks for Qwen3.5-4B pruned with Wanda at 70% global sparsity. Larger profiles indicate better post-pruning performance.

The first level minimizes the full-depth DKNS state cost and coincides with the primary objective in Eq. 4. Among allocations that are optimal at the first level, the second level minimizes the worst normalized depth-prefix ratio. Among allocations tied at the first two levels, the third level favors a smaller layer-wise sparsity variance. We solve this objective using sequential mixed-integer linear programming. The selected states determine the final layer-wise sparsity allocation. Appendix B.2 gives the linear representation of the second-level objective, the sequential solution procedure, and the conditions for strict lexicographic optimality.

3 Experiments

3.1 Experimental Setup

We evaluate NS-UDPD on LLaMA-3/3.1-8B and Qwen3.5-{2B, 4B, 9B}. All experiments are conducted on NVIDIA A100 GPUs. We prune only the prunable Linear modules in Transformer blocks while keeping token embeddings and language modeling heads dense. All methods use the same set of 128 calibration sequences randomly sampled from C4 (Raffel et al., 2020) with random seed 0. At global unstructured sparsity levels of 40%, 50%, 60%, and 70%, we evaluate each layer-wise sparsity allocation under three fixed local pruning rules: Magnitude, Wanda (Sun et al., 2024), and SparseGPT (Frantar & Alistarh, 2023). We compare NS-UDPD with Uniform, OWL (Yin et al., 2024), DLP (Chen et al., 2025), LSA (Yang et al., 2026), and ATP (Huang et al., 2025). Language modeling performance is evaluated using WikiText-2 perplexity (Merity et al., 2017). Zero-shot performance is measured on WinoGrande, HellaSwag, PIQA, OpenBookQA, ARC-Easy, and ARC-Challenge (Sakaguchi et al., 2021; Zellers et al., 2019; Bisk et al., 2020; Mihaylov et al., 2018; Clark et al., 2018). The Mean is recomputed over these six tasks. Section 3.3 presents a joint ablation of the sparsity-grid resolution and the candidate sparsity range. Within each comparison, all allocation methods share the same calibration data, fixed local pruning rule, and target sparsity. Section 3.4 evaluates practical inference speedup using end-to-end throughput measurements.

3.2 Main Results

Language Modeling. Table 1 reports WikiText-2 perplexity (PPL; lower is better) (Merity et al., 2017) for five models from the LLaMA-3 and Qwen3.5 families at 60% and 70% global unstructured sparsity. Within each model–pruner block, all allocation methods follow the same pruning and evaluation protocol. We compare NS-UDPD with Uniform, OWL (Yin et al., 2024), DLP (Chen et al., 2025), LSA (Yang et al., 2026), and ATP (Huang et al., 2025). NS-UDPD achieves the lowest PPL in 24 of the 30 settings, including all ten settings using SparseGPT (Frantar & Alistarh, 2023). Its advantage becomes more pronounced as sparsity increases. On Qwen3.5-2B with SparseGPT, the strongest competing allocations attain PPLs of 29.51 and 64.81 at 60% and 70% sparsity, re-

Table 1: WikiText-2 perplexity (↓) from 50% to 70% global unstructured sparsity. Bold indicates lowest PPL per model–pruner–sparsity setting. Magnitude uses integer rounding for readability.

			Layer-wise Sparsity Allocation					
Sparsity	**Model**	**Pruner**	**Uniform**	**OWL**	**DLP**	**LSA**	**ATP**	**Ours**
70%	LLaMA-3-8B	Magnitude	1,625,607	1,357,574	1,192,404	293,537	121,208	**99,668**
		Wanda	126.03	93.61	108.57	100.58	91.65	**85.78**
		SparseGPT	41.27	37.90	39.48	38.81	37.84	**33.24**
	LLaMA-3.1-8B	Magnitude	1,176,651	2,765,195	64,949	48,944	**22,183**	114,875
		Wanda	110.67	75.18	84.76	75.28	75.14	**75.02**
		SparseGPT	39.29	35.25	39.02	38.32	37.34	**34.11**
	Qwen3.5-2B	Magnitude	1,931,971	770,880	**18,194**	55,954	28,527	74,089
		Wanda	215.55	186.18	242.55	190.19	178.01	**72.48**
		SparseGPT	121.84	82.94	81.28	85.63	64.81	**44.36**
	Qwen3.5-4B	Magnitude	17,379	4,783	28,567	26,355	18,802	**1,022**
		Wanda	98.18	78.19	91.86	73.65	97.26	**38.50**
		SparseGPT	38.06	32.36	35.24	31.72	35.64	**25.85**
	Qwen3.5-9B	Magnitude	264,787	**54,766**	2,773,649	2,436,323	2,825,709	117,266
		Wanda	64.72	51.76	53.69	52.77	54.34	**28.74**
		SparseGPT	26.49	24.46	24.35	23.99	23.78	**19.47**
60%	LLaMA-3-8B	Magnitude	33,919	86,055	60,661	6,647	511	**451**
		Wanda	23.51	18.51	19.00	**17.88**	18.27	22.96
		SparseGPT	15.18	14.71	14.76	14.41	15.11	**14.07**
	LLaMA-3.1-8B	Magnitude	5,334	3,671	5,370	2,140	918	**293**
		Wanda	21.79	17.47	18.21	**17.21**	17.52	23.16
		SparseGPT	15.31	14.52	14.78	14.35	14.80	**14.10**
	Qwen3.5-2B	Magnitude	3,121	2,140	3,444	8,865	4,259	**324**
		Wanda	39.50	36.95	38.63	35.61	40.04	**30.04**
		SparseGPT	32.45	29.51	32.94	31.96	30.71	**21.34**
	Qwen3.5-4B	Magnitude	421	188	220	288	208	**69**
		Wanda	21.45	20.53	23.13	21.19	29.52	**17.33**
		SparseGPT	16.12	15.54	17.09	16.52	19.44	**14.38**
	Qwen3.5-9B	Magnitude	33,205	**3,207**	27,036	69,574	11,713	23,095
		Wanda	14.74	14.06	16.67	16.40	20.23	**12.86**
		SparseGPT	12.84	12.48	14.26	14.09	15.78	**11.56**
50%	LLaMA-3-8B	Magnitude	205.48	215.83	241.54	235.91	177.45	**43.91**
		Wanda	9.84	9.68	9.68	9.63	10.42	**9.52**
		SparseGPT	9.47	9.61	9.39	9.39	10.27	**9.13**
	LLaMA-3.1-8B	Magnitude	134.27	170.92	133.27	134.66	104.50	**31.82**
		Wanda	9.65	9.52	9.55	9.52	10.27	**9.39**
		SparseGPT	9.32	9.43	9.33	9.34	10.25	**9.12**
	Qwen3.5-2B	Magnitude	226.34	131.29	220.26	232.81	251.79	**56.41**
		Wanda	17.95	18.22	17.85	17.83	19.58	**17.53**
		SparseGPT	18.29	17.62	17.47	17.36	18.72	**15.58**
	Qwen3.5-4B	Magnitude	33.69	27.99	31.57	32.26	43.02	**22.52**
		Wanda	12.19	12.31	12.66	12.44	15.47	**11.93**
		SparseGPT	11.50	11.48	11.77	11.65	14.54	**11.19**
	Qwen3.5-9B	Magnitude	1,233.20	210.73	847.06	865.11	178.03	**68.06**
		Wanda	**9.53**	9.76	9.84	9.82	12.45	9.75
		SparseGPT	9.41	9.45	9.73	9.80	11.95	**9.32**

spectively, whereas NS-UDPD obtains 21.34 and 44.36. These results correspond to relative PPL reductions of 27.69% and 31.55%, respectively. Appendix D reports the 40% and 50% results in Table 5, while Fig. 2 shows how the advantage widens across all four sparsity levels.

Table 2: Zero-shot accuracy (%) on six downstream tasks at 70% global unstructured sparsity. Bold indicates the highest six-task Mean within each model–pruner block. Mean is recomputed over the six listed tasks, and OBQA denotes OpenBookQA.

Model	Pruner	Method	WinoGrande	HellaSwag	PIQA	OBQA	ARC-e	ARC-c	Mean
Qwen3.5-2B	Magnitude	Uniform	49.17	26.59	51.25	28.40	25.59	27.65	34.77
		OWL	49.88	26.40	51.31	31.80	26.26	26.02	35.28
		DLP	48.38	27.29	52.50	30.00	26.98	26.37	35.25
		LSA	48.70	26.74	50.76	27.40	24.96	25.60	34.03
		ATP	47.36	27.04	52.77	30.80	26.26	27.39	35.27
		Ours	52.08	26.91	51.82	27.60	26.73	26.92	**35.34**
	Wanda	Uniform	50.83	29.54	55.33	26.20	32.07	22.95	36.15
		OWL	49.41	30.13	57.29	26.20	34.30	20.14	36.24
		DLP	50.83	31.55	58.38	26.80	34.13	22.44	37.35
		LSA	50.28	30.53	57.62	26.00	33.80	21.93	36.69
		ATP	49.41	31.77	58.65	27.20	35.40	22.87	37.55
		Ours	50.75	31.47	59.79	27.40	36.32	20.73	**37.74**
	SparseGPT	Uniform	50.59	30.47	56.91	26.60	34.51	22.18	36.88
		OWL	50.67	32.00	57.24	28.60	35.86	22.95	37.89
		DLP	54.38	33.52	59.19	26.40	35.40	23.38	38.71
		LSA	52.25	32.78	57.83	27.20	34.51	22.61	37.86
		ATP	53.43	33.91	60.12	27.80	36.87	22.70	39.14
		Ours	52.91	33.98	61.17	25.20	39.20	22.90	**39.23**
Qwen3.5-4B	Magnitude	Uniform	50.36	26.79	50.49	28.20	27.61	22.53	34.33
		OWL	52.33	27.36	53.97	27.60	28.70	25.34	35.88
		DLP	50.51	28.45	50.76	29.00	29.76	25.26	35.62
		LSA	49.49	28.07	53.81	26.00	29.12	26.54	35.50
		ATP	49.96	28.72	52.12	30.80	30.51	26.19	36.38
		Ours	50.75	29.63	55.06	27.80	30.93	25.34	**36.59**
	Wanda	Uniform	51.38	31.18	56.69	27.40	35.23	21.33	37.20
		OWL	51.62	32.71	59.09	26.60	37.04	22.27	38.22
		DLP	53.45	36.09	61.99	30.60	39.61	24.97	41.12
		LSA	51.93	36.82	61.48	28.60	39.94	24.74	40.59
		ATP	53.51	36.37	62.79	29.00	39.65	25.17	41.08
		Ours	53.98	36.61	63.40	27.80	41.94	23.80	**41.26**
	SparseGPT	Uniform	54.06	38.20	63.17	26.60	43.64	25.34	41.84
		OWL	54.38	39.98	63.11	28.40	45.50	24.40	42.63
		DLP	58.17	43.09	65.78	28.20	46.89	28.07	45.03
		LSA	56.75	43.09	65.13	28.00	46.38	26.54	44.31
		ATP	56.75	43.21	65.40	29.40	47.43	29.18	**45.23**
		Ours	54.85	40.57	65.23	28.60	46.13	25.85	43.54

Zero-shot Performance. Table 2 reports zero-shot accuracy for Qwen3.5-2B and Qwen3.5-4B at 70% global unstructured sparsity on six downstream tasks (Sakaguchi et al., 2021; Zellers et al., 2019; Bisk et al., 2020; Mihaylov et al., 2018; Clark et al., 2018). The Mean is recomputed over the six tasks. NS-UDPD achieves the highest Mean in five of the six model–pruner blocks; ATP leads the remaining Qwen3.5-4B block with SparseGPT. Relative to Uniform, NS-UDPD raises the Mean from 37.20% to 41.26% on Qwen3.5-4B with Wanda and from 36.88% to 39.23% on Qwen3.5-2B with SparseGPT. These gains indicate that joint layer-wise sparsity selection better preserves downstream capabilities at high sparsity. Fig. 3 provides a task-wise view of the Qwen3.5-4B–Wanda setting, where NS-UDPD exhibits a balanced profile and attains the highest six-task Mean. Appendix E reports the complete results at 60% sparsity.

3.3 Hyperparameter Ablation

NS-UDPD contains two hyperparameters that determine the discrete candidate set: the sparsity-grid resolution Q controls its granularity, while the candidate sparsity range $[s_{\min}, s_{\max}]$ bounds the candidate sparsity levels available to each layer. To examine the influence of these choices, we jointly ablate both hyperparameters using a 3×3 configuration grid. We use Qwen3.5-4B, set SparseGPT as the fixed local pruning rule P, and fix the target mean layer sparsity at $S = 0.70$. We

evaluate WikiText-2 perplexity with $Q \in \{40, 80, 160\}$ and candidate sparsity ranges $[0.60, 0.80]$, $[0.55, 0.85]$, and $[0.50, 0.90]$. All other settings follow the main experiments.

Table 3 shows that NS-UDPD remains stable across the nine configurations, with PPL ranging from 25.85 to 26.45. The intermediate candidate sparsity range $[0.55, 0.85]$ consistently achieves the best or tied-best result. This suggests that a narrow range may restrict useful variation in layer-wise sparsity allocation, whereas further expanding the range introduces more extreme candidate sparsity levels without improving performance. Increasing Q provides no monotonic gain, indicating that a higher sparsity-grid resolution is unnecessary in this setting. The best result is obtained with $Q = 40$ and $[s_{\min}, s_{\max}] = [0.55, 0.85]$. Overall, NS-UDPD is robust to moderate hyperparameter variations and does not rely on a finely tuned discrete candidate set.

Table 3: Joint ablation of the sparsity-grid resolution Q and the candidate sparsity range $[s_{\min}, s_{\max}]$ on Qwen3.5-4B. SparseGPT is used as the fixed local pruning rule P, and the target mean layer sparsity is fixed at $S = 0.70$. We report WikiText-2 perplexity. Bold indicates the best result.

Q	**Candidate sparsity range** $[s_{\min}, s_{\max}]$		
	$[0.60, 0.80]$	$[0.55, 0.85]$	$[0.50, 0.90]$
40	26.45	**25.85**	26.35
80	26.45	26.18	26.35
160	26.22	26.22	26.37

3.4 Inference Speedup

Element-wise unstructured sparsity does not automatically accelerate inference because dense GPU kernels cannot efficiently skip irregularly distributed zeros. To evaluate practical inference efficiency, we apply NS-UDPD to allocate layer-wise sparsity for OPT-6.7B and OPT-13B, with Wanda as the fixed local pruning rule. The resulting models are executed with SpInfer (Fan et al., 2025), a framework optimized for sparse matrix multiplication. All measurements are conducted on NVIDIA A100 40GB GPU with a batch size of 8, an input sequence length of 32, and an output sequence length of 256. Table 4 reports the complete end-to-end throughput and speedup results. As sparsity increases from 50% to 70%, throughput consistently increases for both models, with speedups over their respective dense baselines rising from $1.16\times$–$1.19\times$ to $1.68\times$–$1.69\times$.

Table 4: End-to-end inference throughput (tokens/s) of pruned OPT models. Speedup is measured relative to the corresponding dense model.

Model	**Metric**	**Dense**	**50%**	**60%**	**70%**
OPT-6.7B	Throughput ↑	688.2	819.5	977.2	1163.1
	Speedup ↑	1.00×	1.19×	1.42×	1.69×
OPT-13B	Throughput ↑	397.9	461.6	549.1	668.5
	Speedup ↑	1.00×	1.16×	1.38×	1.68×

4 Conclusion

This paper introduced NS-UDPD, a finite-domain joint selection method for layer-wise sparsity allocation in LLM pruning. DKNS combines squared weights, input-activation second moments, and squared output-NLL-gradient energies into per-weight DKNS proxy deletion costs, which candidate deletion masks aggregate into layer-level DKNS state costs. UDPD controls the accumulation of state costs with depth through prefix-wise comparison with the uniform reference allocation, while lexicographic joint selection optimizes full-depth DKNS state cost, depth-prefix cost distribution, and layer-wise sparsity balance. Experiments across LLaMA-3 and Qwen3.5 series models, fixed local pruning rules, and global unstructured sparsity levels show that NS-UDPD consistently improves model performance, particularly at high sparsity ratios.

Reproducibility Statement

Section 2 presents the complete formulation of NS-UDPD, including the discrete candidate set, DKNS state-cost construction, UDPD constraints, the three-level lexicographic objective, and the sequential mixed-integer linear programming procedure. Appendix Sections A.1–B.2 provide the derivation and interpretation of DKNS, the mathematical properties of layer-wise rebasing and UDPD, and the linear representation and sequential solution of the lexicographic objective. Section 3 specifies the evaluated models, calibration data and random seed, fixed local pruning rules, baselines, target sparsity levels, evaluation tasks and metrics, hardware, and hyperparameter configurations. Appendices D and E report additional language-modeling and zero-shot results, respectively. Together, these descriptions provide the methodological and experimental details needed to reproduce the reported results.

Ethics Statement

This work studies layer-wise sparsity allocation for post-training pruning of publicly available large language models. The study involves no human-subject research and does not involve the collection of new personal or private data. All experiments use publicly released models and established datasets and evaluation benchmarks under their applicable licenses and terms of use. We do not identify additional ethical concerns specific to the proposed method.

AI Use Statement

Generative AI tools were used solely to assist with language polishing and translation. They were not used to develop the core method, design or conduct experiments, or generate or alter experimental measurements. The authors reviewed all AI-assisted text and checked the references against their original publications. The authors take full responsibility for the final manuscript and all claims presented in this work.

REFERENCES

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 10865–10873, 2024.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. Sparsellm: Towards global pruning of pre-trained language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 7432–7439, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Yuli Chen, Bo Cheng, Jiale Han, Yingying Zhang, Yingting Li, and Shuhao Zhang. DLP: dynamic layerwise pruning in large language models. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24:240:1–240:113, 2023.

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. *CoRR*, abs/1803.05457, 2018.

Lucio M. Dery, Steven Kolawole, Jean-François Kagey, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of llms with only forward passes. *CoRR*, abs/2402.05406, 2024.

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 *Proc. International Conference on Machine Learning (ICML)*, pp. 11346–11374, 2024.

Ruibo Fan, Xiangrui Yu, Peijie Dong, Zeyu Li, Gu Gong, Qiang Wang, Wei Wang, and Xiaowen Chu. Spinfer: Leveraging low-level sparsity for efficient large language model inference on gpus. In *Proc. European Conference on Computer Systems (EuroSys)*, pp. 243–260, 2025.

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 *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323, 2022.

Chang Gao, Kang Zhao, Jianfei Chen, and Liping Jing. Maximum redundancy pruning: A principle-driven layerwise sparsity allocation for llms. *CoRR*, abs/2503.18377, 2025.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. DISP-LLM: dimension-independent structural pruning for large language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Wanatnabe. Dependency-aware semi-structured sparsity of GLU variants in large language models. *CoRR*, abs/2405.01943, 2024.

Jiujun He and Huazhen Lin. Olica: Efficient structured pruning of large language models without retraining. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, and Rongrong Ji. Determining layer-wise sparsity for large language models through a theoretical perspective. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

Yazdan Jamshidi and Alexey Shvets. PALS: percentile-aware layerwise sparsity for LLM pruning. *CoRR*, abs/2607.07557, 2026.

Yang Ji and Ying Sun. OCP: outlier-centric probing for dynamic structured pruning of llms. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 6108–6124, 2026.

Khang Nguyen Le, Ryo Sato, Dai Nakashima, Takeshi Suzuki, and Minh Le Nguyen. Optiprune: Effective pruning approach for every target sparsity. In *Proc. International Conference on Computational Linguistics (COLING)*, pp. 3600–3612, 2025.

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 *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024a.

Wei Li, Lujun Li, Mark G. Lee, and Shengjie Sun. Adaptive layer sparsity for large language models via activation correlation assessment. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024b.

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 *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Mingge Lu, Jingwei Sun, Junqing Lin, Zechun Zhou, and Guangzhong Sun. Lua-llm: Learning unstructured-sparsity allocation for large language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proc. International Conference on Machine Learning (ICML)*, pp. 2408–2417, 2015.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 20192–20204, 2025.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. ALPS: improved optimization for highly sparse one-shot pruning for large language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024a.

Xiang Meng, Shibal Ibrahim, Kayhan Behdin, Hussein Hazimeh, Natalia Ponomareva, and Rahul Mazumder. OSSCAR: one-shot structured pruning in vision and language models with combinatorial optimization. In *Proc. International Conference on Machine Learning (ICML)*, pp. 35354–35377, 2024b.

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

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 *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2381–2391, 2018.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

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

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.

Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. Search for efficient large language models. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: streamlining llms through redundancy verification and elimination of transformer blocks. In *Proc. International Conference on Machine Learning (ICML)*, pp. 46136–46155, 2024.

Lu Sun and Jun Sakuma. Learning semi-structured sparsity for llms via shared and context-aware hypernetwork. In *Proc. International Conference on Learning Representations (ICLR)*, 2026.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proc. International Conference on Machine Learning (ICML)*, pp. 38087–38099, 2023.

Lang Xiong, Ning Liu, Ao Ren, Yuheng Bai, Haining Fang, Binyan Zhang, Zhe Jiang, Yujuan Tan, and Duo Liu. D2 prune: Sparsifying large language models via dual taylor expansion and attention distribution awareness. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 27171–27179, 2026.

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 *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Mingzhe Yang, Sihao Lin, Changlin Li, and Xiaojun Chang. Let LLM tell what to prune and how much to prune. In *Proc. International Conference on Machine Learning (ICML)*, 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 *Proc. International Conference on Learning Representations (ICLR)*, 2026.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

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 *Proc. International Conference on Machine Learning (ICML)*, pp. 57101–57115, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800, 2019.

Stephen Zhang and Vardan Papyan. OATS: outlier-aware pruning through sparse and low rank decomposition. In *Proc. International Conference on Learning Representations (ICLR)*, 2025.

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 *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Pengxiang Zhao, Hanyu Hu, Ping Li, Yi Zheng, Zhefeng Wang, and Xiaoming Yuan. Fistapruner: Layer-wise post-training pruning for large language models. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 29469–29487, 2025.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5065–5080, 2025.

Changhai Zhou, Qian Qiao, Weizhong Zhang, and Cheng Jin. Large language model compression with global rank and sparsity optimization. *CoRR*, abs/2505.03801, 2025.

A Derivation and Mathematical Interpretation of DKNS

A.1 From Shared-Weight Gradients to the Diagonal-Kronecker Approximation

Consider a single prunable Linear module and omit its layer and module indices. Define $g_{n,q,i} = \partial \mathcal{L}_n / \partial y_{n,q,i}$. Since the same weight matrix is shared across all valid positions in a sequence, the sequence-level gradient of $w_{i,j}$ is

$$\frac{\partial \mathcal{L}_n}{\partial w_{i,j}} = \sum_{q \in \mathcal{I}_n} g_{n,q,i} x_{n,q,j}. \tag{A.1}$$

Consider the uncentered empirical second moment of the sequence gradients. Its entries can be expanded as

$$\begin{aligned} F^{\text{emp}}_{(i,j),(i',j')} &= \frac{1}{N} \sum_{n=1}^{N} \frac{\partial \mathcal{L}_n}{\partial w_{i,j}} \frac{\partial \mathcal{L}_n}{\partial w_{i',j'}} \\ &= \frac{1}{N} \sum_{n=1}^{N} \sum_{q \in \mathcal{I}_n} g_{n,q,i} g_{n,q,i'} x_{n,q,j} x_{n,q,j'} \\ &\quad + \frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{q,q' \in \mathcal{I}_n \\ q \neq q'}} g_{n,q,i} g_{n,q',i'} x_{n,q,j} x_{n,q',j'}. \end{aligned} \tag{A.2}$$

The first term contains same-position contributions, whereas the second contains cross-position correlations induced by weight sharing. DKNS first retains only the diagonal parameter coordinates and discards their cross-position terms. The retained same-position contribution is

$$f^{\text{pos}}_{i,j} = \frac{1}{N} \sum_{n=1}^{N} \sum_{q \in \mathcal{I}_n} g^2_{n,q,i} x^2_{n,q,j}. \tag{A.3}$$

Let $\nu = \sum_{n=1}^{N} |\mathcal{I}_n|$ and $\bar{\nu} = \nu / N$, and let π denote the uniform distribution over all ν valid positions. The statistics in Eq. 5 satisfy $\widehat{A}_j = \mathbb{E}_\pi[x_j^2]$ and $\widehat{G}_i = \bar{\nu}\, \mathbb{E}_\pi[g_i^2]$. DKNS further factorizes the joint empirical moment of the squared output gradient and squared input activation at the same position:

$$\begin{aligned} f^{\text{pos}}_{i,j} &= \bar{\nu}\, \mathbb{E}_\pi[g_i^2 x_j^2] \\ &\approx \bar{\nu}\, \mathbb{E}_\pi[g_i^2] \mathbb{E}_\pi[x_j^2] \\ &= \widehat{G}_i \widehat{A}_j. \end{aligned} \tag{A.4}$$

When the parameter coordinates are ordered as (j, i), these coordinate-wise coefficients correspond to $D^A \otimes D^G$ in Eq. 6. A different coordinate ordering applies only a common permutation to the rows and columns. It leaves the scalar quadratic form in Eq. 7 and the per-weight costs in Eq. 8 unchanged. DKNS therefore involves three approximations: retaining only the diagonal of the empirical gradient second-moment matrix, discarding the cross-position terms induced by weight sharing, and factorizing the same-position joint moment of squared output gradients and squared input activations. The discarded cross-position terms can be positive or negative, so removing them does not generally yield an upper or lower bound. DKNS is a tractable proxy motivated by empirical gradient second-order statistics. It is not the full empirical Fisher, the true Hessian, or a general equivalence between them.

A.2 Relation to Taylor Curvature, Euclidean Reconstruction Error, and Additivity

Let $\mathcal{L}(W) = N^{-1}\sum_{n=1}^{N}\mathcal{L}_n(W)$ denote the calibration objective as a function of W, with all other model parameters fixed. Its local Taylor expansion is

$$\begin{aligned}\mathcal{L}(W+\Delta W) - \mathcal{L}(W) = &\langle \nabla_W \mathcal{L}, \Delta W\rangle_F \\ &+ \frac{1}{2}\langle \Delta W, H_W[\Delta W]\rangle_F \\ &+ o\big(\|\Delta W\|_F^2\big).\end{aligned} \tag{A.5}$$

Here, $H_W[\Delta W]$ denotes the action of the NLL Hessian on the perturbation ΔW. DKNS does not approximate the complete signed loss change in Eq. (A.5). It also does not assume that the first-order term vanishes or that the empirical gradient second moment equals the true Hessian. Instead, it defines the following nonnegative local quadratic sensitivity proxy:

$$\mathcal{S}^{\mathrm{DKNS}}(\Delta W) = \frac{1}{2}\sum_{i,j}\widehat{G}_i\widehat{A}_j\,(\Delta w_{i,j})^2 \geq 0. \tag{A.6}$$

DKNS should therefore be interpreted as an NLL-related local perturbation sensitivity, rather than the exact pruning-induced NLL change or a general upper bound on that change. Define the full input second moment along the dense-model calibration trajectory as $A^{\mathrm{full}} = \nu^{-1}\sum_{n=1}^{N}\sum_{q\in\mathcal{I}_n} x_{n,q}x_{n,q}^{\top}$. With the input activations fixed, the local Euclidean output-reconstruction error is

$$\begin{aligned}\mathcal{E}_{\mathrm{Euc}}(\Delta W) &= \frac{1}{2\nu}\sum_{n=1}^{N}\sum_{q\in\mathcal{I}_n}\|\Delta W x_{n,q}\|_2^2 \\ &= \frac{1}{2}\,\mathrm{tr}\left(\Delta W A^{\mathrm{full}}\Delta W^{\top}\right) \\ &\approx \frac{1}{2}\sum_{i,j}\widehat{A}_j\,(\Delta w_{i,j})^2.\end{aligned} \tag{A.7}$$

The final line uses a diagonal approximation to the input second moment. This Euclidean proxy uses the identity metric over the output space, whereas DKNS weights different output channels by $\widehat{G}_i$.

For a deletion set Ω, define the weight-zeroing perturbation by $\Delta w_{i,j}^{\Omega} = -w_{i,j}$ when $(i,j)\in\Omega$ and $\Delta w_{i,j}^{\Omega} = 0$ otherwise. Since DKNS is a diagonal quadratic form,

$$\mathcal{S}^{\mathrm{DKNS}}(\Delta W^{\Omega}) = \sum_{(i,j)\in\Omega} d_{i,j}^{\mathrm{DKNS}}. \tag{A.8}$$

Thus, within a single Linear module and the defined DKNS proxy, the cost of a deletion set is exactly the sum of the corresponding per-weight proxy deletion costs. Equation (9) further aggregates these costs across Linear modules. This aggregation is a separable modeling assumption used by the finite-domain joint selection problem, rather than an exact decomposition of the end-to-end NLL change. If the local pruning rule also applies compensation updates to the retained weights, let $\Delta w_{i,j}^{\mathrm{comp}}$ denote the update at $(i,j)\notin\Omega$. The DKNS proxy for the complete perturbation is

$$\mathcal{S}^{\mathrm{DKNS}}(\Delta W^{\mathrm{full}}) = \sum_{(i,j)\in\Omega} d_{i,j}^{\mathrm{DKNS}} + \frac{1}{2}\sum_{(i,j)\notin\Omega}\widehat{G}_i\widehat{A}_j\left(\Delta w_{i,j}^{\mathrm{comp}}\right)^2. \tag{A.9}$$

The state cost in Eq. 9 retains only the first term to evaluate the deletion set determined by each candidate state. It does not claim to exactly measure compensation updates to the retained weights or the end-to-end NLL change of the complete sparse model.

B Mathematical Properties of UDPD and Lexicographic Joint Selection

B.1 Invariance of Layer-Wise Rebasing and Feasibility of the Uniform Reference Allocation

Define the minimum raw state cost of layer ℓ as $b_\ell^{(P)}$, and let B_d denote its cumulative offset up to depth d. Let $\widetilde{A}_d(x)$ and $\widetilde{U}_d$ denote the cumulative state costs of the current allocation and the uniform reference allocation before rebasing, respectively:

$$\begin{aligned} b_\ell^{(P)} &= \min_{u\in\mathcal{T}} \widetilde{c}_{\ell,u}^{(P)}, & B_d &= \sum_{\ell=1}^{d} b_\ell^{(P)}, \\ \widetilde{A}_d(x) &= \sum_{\ell=1}^{d}\sum_{t\in\mathcal{T}} \widetilde{c}_{\ell,t}^{(P)} x_{\ell,t}, & \widetilde{U}_d &= \sum_{\ell=1}^{d} \widetilde{c}_{\ell,t^\star}^{(P)}. \end{aligned} \tag{B.1}$$

Because exactly one state is selected in each layer, Eq. (11) gives

$$A_d(x) = \widetilde{A}_d(x) - B_d, \qquad U_d = \widetilde{U}_d - B_d. \tag{B.2}$$

Therefore, for any two allocations x and y satisfying the one-state-per-layer constraint,

$$\begin{aligned} A_d(x) - A_d(y) &= \widetilde{A}_d(x) - \widetilde{A}_d(y), \\ A_d(x) - U_d &= \widetilde{A}_d(x) - \widetilde{U}_d. \end{aligned} \tag{B.3}$$

Thus, rebasing preserves allocation cost differences at every depth, full-depth ordering at $d = L$, and unnormalized prefix-cost differences from the uniform reference; hence, UDPD feasibility is unchanged. Since this invariance does not apply to $A_d(x)/U_d$, the second-level objective uses rebased costs. With $s_t = t/Q$, the uniform reference allocation is defined as

$$x_{\ell,t}^{\mathrm{uni}} = \mathbf{1}\{t = t^\star\}, \qquad \ell = 1,\dots,L, \quad t \in \mathcal{T}. \tag{B.4}$$

This allocation satisfies the binary constraints and the one-state-per-layer constraint. It also satisfies

$$\frac{1}{L}\sum_{\ell=1}^{L}\sum_{t\in\mathcal{T}} s_t x_{\ell,t}^{\mathrm{uni}} = \frac{t^\star}{Q} = S. \tag{B.5}$$

Furthermore,

$$A_d\big(x^{\mathrm{uni}}\big) = U_d, \qquad d = 1,\dots,L. \tag{B.6}$$

Thus, $x^{\mathrm{uni}} \in \mathcal{X}_{\mathrm{UDPD}}$, proving that the UDPD feasible set is nonempty. Equation (11) also guarantees $c_{\ell,t}^{(P)} \geq 0$. For any UDPD-feasible allocation, if $U_d = 0$, then

$$0 \leq A_d(x) \leq U_d = 0 \quad \Longrightarrow \quad A_d(x) = 0. \tag{B.7}$$

Prefixes with $U_d = 0$ are therefore excluded from the normalized ratio. If $\mathcal{D}_+ =$, the second-level objective is identically zero, and the corresponding optimization stage is omitted.

B.2 Linear Representation and Sequential Solution of the Lexicographic Objective

For $x \in \mathcal{X}_{\mathrm{UDPD}}$, define

$$R(x) = \begin{cases} \max\limits_{d\in\mathcal{D}_+} \dfrac{A_d(x)}{U_d}, & \mathcal{D}_+ \neq, \\ 0, & \mathcal{D}_+ = . \end{cases} \tag{B.8}$$

When $\mathcal{D}_+ \neq$, introduce a continuous variable ρ. For any fixed $x \in \mathcal{X}_{\mathrm{UDPD}}$, we have

$$R(x) = \min_{\rho} \{\rho : A_d(x) \leq \rho U_d \text{ for all } d \in \mathcal{D}_+,\ 0 \leq \rho \leq 1\}. \tag{B.9}$$

The bounds on ρ follow because the rebased state costs are nonnegative and UDPD bounds every included ratio by one. Each U_d is constant, and $A_d(x)$ is linear in x. Equation (B.9) therefore provides a linear representation of the second-level objective. The selected state index and the corresponding sparsity level of layer ℓ are

$$t_\ell(x) = \sum_{t \in \mathcal{T}} t\, x_{\ell,t}, \qquad s_\ell(x) = \frac{t_\ell(x)}{Q}. \tag{B.10}$$

The exact mean layer-sparsity constraint gives

$$\frac{1}{L}\sum_{\ell=1}^{L} t_\ell(x) = t^\star, \qquad \frac{1}{L}\sum_{\ell=1}^{L} s_\ell(x) = S. \tag{B.11}$$

Because exactly one binary state is selected in each layer, the third-level objective can be written as

$$\begin{aligned} V(x) &= \sum_{\ell=1}^{L}\sum_{t \in \mathcal{T}} (t - t^\star)^2 x_{\ell,t} \\ &= \sum_{\ell=1}^{L} (t_\ell(x) - t^\star)^2 \\ &= LQ^2 \left[\frac{1}{L}\sum_{\ell=1}^{L} (s_\ell(x) - S)^2\right]. \end{aligned} \tag{B.12}$$

The bracketed term is the equally weighted variance of the layer-wise sparsity levels. Therefore, among allocations tied at the first two levels, minimizing $V(x)$ is equivalent to minimizing the layer-wise sparsity variance. Let $\epsilon_1, \epsilon_2 \geq 0$ denote the stage-locking tolerances for the first two objectives. The first stage solves

$$\phi_1^\star = \min_{x \in \mathcal{X}_{\mathrm{UDPD}}} A_L(x). \tag{B.13}$$

When $\mathcal{D}_+ \neq$, the second stage solves

$$\begin{aligned} \phi_2^\star = \min_{x,\rho}\ & \rho \\ \text{s.t.}\ & x \in \mathcal{X}_{\mathrm{UDPD}}, \\ & A_L(x) \leq \phi_1^\star + \epsilon_1, \\ & A_d(x) \leq \rho U_d, \qquad d \in \mathcal{D}_+, \\ & 0 \leq \rho \leq 1. \end{aligned} \tag{B.14}$$

The third stage solves

$$\begin{aligned} \min_{x,\rho}\ & V(x) \\ \text{s.t.}\ & x \in \mathcal{X}_{\mathrm{UDPD}}, \\ & A_L(x) \leq \phi_1^\star + \epsilon_1, \\ & A_d(x) \leq \rho U_d, \qquad d \in \mathcal{D}_+, \\ & 0 \leq \rho \leq 1, \\ & \rho \leq \phi_2^\star + \epsilon_2. \end{aligned} \tag{B.15}$$

Let $x^\star$ denote the final allocation obtained in the third stage. If $\mathcal{D}_+ =$, the second stage is omitted, and ρ and its associated constraints are removed from the third stage, which minimizes $V(x)$ after locking the first-level objective. When all stages are solved exactly and $\epsilon_1 = \epsilon_2 = 0$, each stage is restricted to the exact optimal set preserved by all preceding stages. Consequently, $x^\star$ satisfies the strict lexicographic optimality in Eq. 14 without requiring a unique solution. With nonzero stage-locking or solver tolerances, this guarantee holds only up to the corresponding tolerances.

C Related Work

C.1 LLM Pruning

Pruning reduces the storage and computational costs of large language models (LLMs) by removing redundant weights or structures, and the prohibitive cost of full retraining has shifted recent work toward calibration-only post-training pruning at unstructured, semi-structured, and structured granularities (Frantar & Alistarh, 2023; Sun et al., 2024; Ma et al., 2023). At the weight level, SparseGPT solves layer-wise sparse reconstruction with second-order information and compensatory weight updates (Frantar & Alistarh, 2023), whereas Wanda scores each weight by its magnitude and input-activation norm without updating the retained parameters (Sun et al., 2024). RIA normalizes weight magnitude across input and output channels before incorporating activations (Zhang et al., 2024), while SparseLLM decomposes global pruning into coordinated local subproblems through auxiliary variables (Bai et al., 2024). Beyond hand-crafted saliency, Pruner-Zero evolves symbolic metrics (Dong et al., 2024), ALPS combines operator splitting with conjugate-gradient post-processing (Meng et al., 2024a), and FISTAPruner solves a LASSO-like convex objective using FISTA (Zhao et al., 2025). OATS preserves input-feature outliers through activation-aware sparse–low-rank weight decomposition (Zhang & Papyan, 2025), whereas D2Prune jointly models weight and activation perturbations and preserves long-tail attention distributions (Xiong et al., 2026). For hardware-compatible $(N : M)$ sparsity, MaskLLM learns mask distributions through Gumbel-Softmax (Fang et al., 2024), while DaSS incorporates GLU dependencies and intermediate-activation norms (Guo et al., 2024). At coarser granularity, OSSCAR optimizes one-shot neuron or head removal combinatorially (Meng et al., 2024b), LLM-Pruner scores non-critical coupled structures with gradients (Ma et al., 2023), SliceGPT exploits computational invariance to remove embedding-dimension rows and columns (Ashkboos et al., 2024), and FLAP standardizes fluctuation-based channel scores with bias compensation (An et al., 2024). DISP-LLM eliminates the structural dependence along the embedding dimension (Gao et al., 2024), while Olica compresses coupled MHA products through orthogonal neuron decomposition and repairs FFN residual errors by low-rank linear calibration (He & Lin, 2025). ShortGPT and SLEB remove entire layers or blocks (Men et al., 2025; Song et al., 2024), BlockPruner searches over MHA and MLP blocks (Zhong et al., 2025), and Bonsai estimates structured importance through forward-only perturbations (Dery et al., 2024). Search-LLM further casts pruning as training-free architecture search followed by calibration-based weight reconstruction (Shen et al., 2024). Collectively, these methods mainly optimize what to prune, whereas layer-wise sparsity allocation determines how much to prune each layer under a fixed global budget according to its pruning sensitivity.

C.2 Layer-wise Sparsity Allocation for LLM Pruning

Layer-wise sparsity allocation addresses this budget decision before local masks are generated, motivated by the fact that uniform ratios used by one-shot baselines ignore large differences in layer sensitivity. Early proxy-based methods translate layer heterogeneity into explicit schedules. OWL protects layers containing more activation-aware weight outliers Yin et al. (2024), whereas AlphaPruning assigns lower sparsity to better-trained layers using heavy-tailed empirical spectral-density shapes Lu et al. (2024). ALS measures inter-layer dependence through information orthogonality and solves the resulting linear program Li et al. (2024b), while DLP uses the median of activation-aware weight scores to avoid manually tuned outlier thresholds Chen et al. (2025). Optimization and search subsequently reduce reliance on fixed proxy-to-schedule rules. BESA learns differentiable mixtures of candidate sparsity ratios under a block-wise reconstruction objective Xu et al. (2024), whereas DSA evolves symbolic allocation functions from element-wise pruning scores to layer-wise sparsity ratios Li et al. (2024a). ATP derives a monotonically increasing arithmetic schedule from the cross-layer propagation of reconstruction errors Huang et al. (2025), while MRP iteratively prunes the most redundant layers to promote uniform post-pruning redundancy across layers Gao et al. (2025). OptiPrune adaptively adjusts both the deviation from uniform sparsity allocation and the choice between RIA and SparseGPT according to the target sparsity Le et al. (2025), whereas PALS confines adjustments based on the 99th activation percentile to $\pm 5\%$ around the target ratio Jamshidi & Shvets (2026). Recent methods further increase allocation granularity: Lua-LLM learns global inter- and intra-layer sparsity allocation through soft Top-K-based mask optimization Lu et al. (2025), while LSA estimates minimal linear reconstruction error and supports block-, and projection-level allocation Yang et al. (2026). The same allocation principle now extends

beyond static unstructured weight pruning: Yang et al. allocate pruning budgets across heterogeneous structural units using transfer entropy and information entropy Yang et al. (2025), HyperPrune generates layer- and component-conditioned $(N:M)$ masks using a shared context-aware hypernetwork Sun & Sakuma (2026), and OCP adjusts structured sparsity online using history-guided outlier statistics Ji & Sun (2026). CAP jointly allocates sparse and low-rank resources after robust-PCA decomposition, extending sparsity allocation from pruning ratios to coordinated rank-and-sparsity optimization Zhou et al. (2025). Overall, the field has progressed from proxy-based allocation to optimization, automated search, fine-grained learning, and adaptive coordination. Yet many methods still rely on surrogate statistics or local reconstruction objectives, leaving model-wide functional damage from actual pruning underexplored.

D LANGUAGE MODELING AT MODERATE SPARSITY

Table 5 complements the high-sparsity results in Table 1 by reporting WikiText-2 perplexity at 40% and 50% global unstructured sparsity. NS-UDPD achieves the lowest PPL in 27 of the 30 model–pruner–sparsity settings, including all ten settings using Magnitude, eight of ten using Wanda, and nine of ten using SparseGPT. In the remaining three settings, its gap from the best allocation is at most 0.22 PPL. Together with the results at 60% and 70% sparsity, these findings show that NS-UDPD remains effective at moderate sparsity, while its advantage becomes more pronounced at high sparsity levels.

Table 5: WikiText-2 perplexity (↓) at 40% and 50% global unstructured sparsity. Bold indicates lowest PPL per model–pruner–sparsity setting. All results are reported to two decimal places.

			Layer-wise Sparsity Allocation					
Sparsity	**Model**	**Pruner**	**Uniform**	**OWL**	**DLP**	**LSA**	**ATP**	**Ours**
50%	LLaMA-3-8B	Magnitude	205.48	215.83	241.54	235.91	177.45	**43.91**
		Wanda	9.84	9.68	9.68	9.63	10.42	**9.52**
		SparseGPT	9.47	9.61	9.39	9.39	10.27	**9.13**
	LLaMA-3.1-8B	Magnitude	134.27	170.92	133.27	134.66	104.50	**31.82**
		Wanda	9.65	9.52	9.55	9.52	10.27	**9.39**
		SparseGPT	9.32	9.43	9.33	9.34	10.25	**9.12**
	Qwen3.5-2B	Magnitude	226.34	131.29	220.26	232.81	251.79	**56.41**
		Wanda	17.95	18.22	17.85	17.83	19.58	**17.53**
		SparseGPT	18.29	17.62	17.47	17.36	18.72	**15.58**
	Qwen3.5-4B	Magnitude	33.69	27.99	31.57	32.26	43.02	**22.52**
		Wanda	12.19	12.31	12.66	12.44	15.47	**11.93**
		SparseGPT	11.50	11.48	11.77	11.65	14.54	**11.19**
	Qwen3.5-9B	Magnitude	1,233.20	210.73	847.06	865.11	178.03	**68.06**
		Wanda	**9.53**	9.76	9.84	9.82	12.45	9.75
		SparseGPT	9.41	9.45	9.73	9.80	11.95	**9.32**
40%	LLaMA-3-8B	Magnitude	43.25	85.00	51.91	50.57	104.91	**11.39**
		Wanda	7.44	7.60	7.46	7.48	8.02	**7.41**
		SparseGPT	7.49	7.75	7.55	7.56	8.21	**7.44**
	LLaMA-3.1-8B	Magnitude	19.64	36.50	23.02	22.64	64.06	**10.73**
		Wanda	7.49	7.66	**7.49**	7.51	8.06	7.54
		SparseGPT	7.52	7.73	7.57	7.58	8.13	**7.46**
	Qwen3.5-2B	Magnitude	44.06	39.68	41.46	41.62	34.09	**23.47**
		Wanda	13.94	14.06	14.23	14.17	14.87	**13.88**
		SparseGPT	13.83	13.79	14.05	13.90	14.66	**13.35**
	Qwen3.5-4B	Magnitude	14.71	14.72	14.99	14.98	19.00	**13.67**
		Wanda	10.44	10.37	10.57	10.51	11.82	**10.35**
		SparseGPT	10.39	10.28	10.38	10.35	11.85	**10.15**
	Qwen3.5-9B	Magnitude	80.88	17.72	69.83	81.73	20.65	**15.68**
		Wanda	8.61	8.67	8.79	8.80	10.01	**8.55**
		SparseGPT	8.63	**8.58**	8.70	8.75	9.97	8.62

E Additional Zero-Shot Results

Table 6 reports the complete task-wise results at 60% global unstructured sparsity. NS-UDPD achieves the highest six-task Mean in four of the six model–pruner blocks and outperforms Uniform in all six. The largest gains over Uniform are 4.54 percentage points on Qwen3.5-4B with Magnitude, from 38.57% to 43.11%, and 3.96 points on Qwen3.5-2B with SparseGPT, from 41.99% to 45.95%. Together with the 70% results, these findings show that NS-UDPD generalizes across pruning rules, although the relative ranking of allocation methods can depend on the final pruner.

Table 6: Zero-shot accuracy (%) on six downstream tasks at 60% global unstructured sparsity. Bold indicates the highest six-task Mean within each model–pruner block. Mean is recomputed over the six listed tasks, and OBQA denotes OpenBookQA.

Model	Pruner	Method	WinoGrande	HellaSwag	PIQA	OBQA	ARC-e	ARC-c	Mean
Qwen3.5-2B	Magnitude	Uniform	49.49	26.73	52.61	29.60	29.21	25.34	35.50
		OWL	51.62	28.38	53.75	25.80	28.96	24.06	35.43
		DLP	49.33	27.53	52.83	25.60	29.34	24.66	34.88
		LSA	51.54	26.73	52.83	27.00	28.66	24.23	35.16
		ATP	49.49	28.15	54.73	27.80	28.45	23.72	35.39
		Ours	52.80	29.04	57.02	27.20	31.36	23.63	**36.84**
	Wanda	Uniform	54.54	38.54	63.06	26.20	44.49	24.32	41.86
		OWL	53.67	39.56	63.76	27.60	46.00	25.94	42.76
		DLP	56.43	41.02	63.60	29.60	45.79	25.60	**43.67**
		LSA	55.56	40.56	63.11	27.80	44.40	25.51	42.83
		ATP	55.01	40.98	64.36	29.20	44.11	26.19	43.31
		Ours	54.93	39.38	63.38	28.60	46.17	25.00	42.91
	SparseGPT	Uniform	51.70	40.02	63.60	28.40	40.99	27.22	41.99
		OWL	54.46	42.44	65.78	29.00	46.59	30.20	44.75
		DLP	56.35	42.07	65.51	28.20	45.03	29.01	44.36
		LSA	55.64	41.15	64.64	28.20	44.53	28.84	43.83
		ATP	56.75	44.40	67.36	29.40	47.77	29.44	45.85
		Ours	56.04	43.46	67.93	29.80	49.13	29.32	**45.95**
Qwen3.5-4B	Magnitude	Uniform	51.22	33.18	56.58	26.60	37.54	26.28	38.57
		OWL	50.04	37.50	60.66	28.00	42.34	29.78	41.39
		DLP	50.20	39.26	60.23	28.60	43.90	29.44	41.94
		LSA	48.54	37.51	60.61	29.00	41.62	27.65	40.82
		ATP	52.88	40.30	62.68	28.20	43.31	29.10	42.74
		Ours	52.64	40.81	63.49	27.80	44.57	29.35	**43.11**
	Wanda	Uniform	59.04	49.92	67.95	31.60	59.09	34.90	50.42
		OWL	58.64	49.99	69.15	31.80	61.70	36.26	51.26
		DLP	60.62	52.22	70.62	32.60	62.79	37.88	**52.79**
		LSA	62.19	52.43	70.08	32.80	60.23	36.43	52.36
		ATP	59.19	50.81	70.84	32.40	61.32	38.31	52.15
		Ours	59.72	52.16	71.89	32.20	60.18	35.64	51.97
	SparseGPT	Uniform	61.96	56.41	71.65	34.60	63.47	40.53	54.77
		OWL	61.96	56.08	71.55	34.60	64.23	39.93	54.72
		DLP	62.04	57.05	71.06	32.60	65.07	41.30	54.85
		LSA	62.75	**57.55**	70.78	33.00	63.68	40.19	54.66
		ATP	63.06	55.89	71.65	33.00	64.10	38.99	54.45
		Ours	62.78	56.67	72.09	34.60	64.16	39.61	**54.99**