

PerturbPrune: Guiding Sparsity Allocation by Probing Layer Stability Along Principal Directions

Ningxin Wu^{1*} Zhengxu Hou^{3*} Ziran Zhao¹ Tao Jing¹
 Pengyuan Liu^{1,2} Chen Kang¹
 Yangshijie Zhang³ Luxi Xing³ Bin-Bin Yang³ Feng Xiao³ Lu Liu³ Dong Yu^{1†}

¹School of Information Science, Beijing Language and Culture University, Beijing, China

²National Print Media Language Resources Monitoring & Research Center, Beijing, China

³Alibaba Group

Abstract

Large language models (LLMs) exhibit significant layer-wise redundancy, motivating non-uniform pruning for efficient compression. However, existing sparsity allocation methods rely on indirect statistical heuristics derived from weight magnitudes or activation distributions, without directly assessing how robustly each layer preserves semantic information under input variation. We propose PerturbPrune, a framework that allocates layer-wise sparsity by probing the intrinsic stability of hidden representations. PerturbPrune estimates the principal subspace of each layer’s activations via PCA and injects structured perturbations along these directions. Layer sensitivity is quantified through matrix difference entropy (DiME): stable layers are identified as encoding compact, essential features and thus preserved, while highly sensitive layers are deemed over-parameterized and pruned more aggressively. Extensive experiments across diverse LLM families (LLaMA, LLaMA-2, Qwen, OPT) demonstrate that PerturbPrune broadly matches or outperforms existing non-uniform methods. Compared to uniform pruning, it prevents catastrophic collapse, reducing perplexity by up to 99.7% and improving zero-shot accuracy by up to 22.7%. The method is also highly efficient, completing sparsity allocation on Qwen3.5-9B in 7 minutes—achieving 24× and 17.4× speedups over AlphaPruning and MRP. Finally, PerturbPrune generalizes to Mixture-of-Experts architectures, achieving up to 19.3% perplexity reduction on Qwen3.5-35B-A3B.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Touvron et al., 2023a; Grattafiori et al., 2024) have demonstrated strong performance across text

* Equal contribution. 202421198430@stu.blcu.edu.cn, houzhengxu.hzx@alibaba-inc.com

† Corresponding author: yudong@blcu.edu.cn

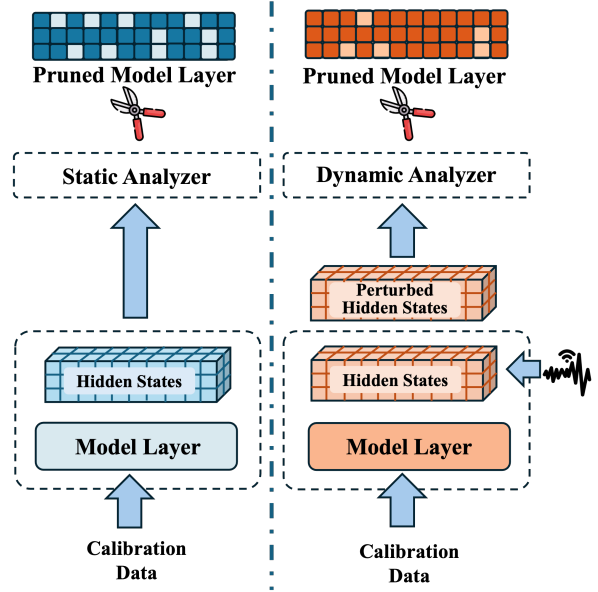


Figure 1: Overview of two sparsity allocation paradigms. (Left) Existing methods estimate layer importance from static representation statistics. (Right) PerturbPrune injects structured noise into hidden states, measuring representational stability via output-to-input deviation ratio. Lighter blocks denote pruned weights.

generation, complex reasoning, and semantic understanding (Chowdhery et al., 2023; Li et al., 2025b), with model scaling consistently improving task performance (Wei et al., 2022). However, the massive computational cost and memory footprint pose significant bottlenecks for deployment, particularly on resource-constrained platforms, making effective model compression urgently needed. Weight pruning (Frantar and Alistarh, 2023; Sun et al., 2023) is highly effective for reducing redundancy and accelerating inference by inducing sparsity. However, early pruning methods apply uniform sparsity ratios across all Transformer layers, which is suboptimal given the heterogeneous sensitivity of LLM layers to parameter removal, motivating the need for layer-aware non-uniform

pruning strategies.

Recent non-uniform sparsity allocation methods for LLM pruning rely primarily on layer-wise importance heuristics derived from static weight or activation statistics. OWL (Yin et al., 2023) uses magnitude-based outlier statistics as a proxy for importance, while AlphaPruning (Lu et al., 2024) exploits heavy-tailed weight distributions but requires over 5 hours to compute these distributions for 32B models. MRP (Gao et al., 2025) allocates sparsity based on layer-wise redundancy measured by singular value decomposition, but requires iterative re-estimation of redundancy across layers, incurring substantial computational overhead (up to 1 day for 32B models), and relies on heuristic step-size decay strategies that are prone to local optima. Although DSA (Li et al., 2024) avoids handcrafted heuristics via evolutionary search, it is computationally expensive and lacks interpretability.

Static approaches are fundamentally limited by their use of snapshot statistics that ignore how layers respond to input variations. While these metrics capture parameter scales, they fail to assess representational stability. For instance, a layer with large-norm weights might be highly redundant, while another with modest statistics could be functionally critical (See Figure 7 Layer 0). Such a mismatch is particularly problematic under extreme sparsity, where misidentifying critical layers leads to catastrophic degradation. Therefore, a dynamic assessment of layer response to input variations is essential.

We address this limitation by reframing layer importance through dynamic representational stability (Figure 1). Instead of relying on static statistics, our approach evaluates each layer’s functional role by probing its response to controlled perturbations within the representation space. Layers that maintain stable outputs under perturbation are deemed robust and preserved, while layers that amplify perturbations exhibit fragility and receive higher sparsity ratios. This stability-based criterion provides a direct proxy for a layer’s contribution to representational integrity, enabling principled non-uniform sparsity allocation.

We introduce **PerturbPrune**, a framework that operationalizes this stability principle for non-uniform sparsity allocation. **PerturbPrune** begins by collecting hidden states from a small calibration set. For each layer, it applies PCA to these states to identify a low-dimensional subspace that cap-

tures their most salient representational directions. Two independent perturbations are then generated within this subspace and added back to the original hidden states. After passing each perturbed version through the layer, **PerturbPrune** measures the matrix difference entropy via DiME (Skean et al., 2023) between the resulting outputs. A high DiME score signifies high stability, indicating the layer effectively absorbs variations and maintains consistent outputs, thus warranting a lower sparsity ratio. Conversely, a low DiME score reveals fragility, permitting more aggressive pruning. These stability scores are then mapped to final layer-wise sparsity ratios via a monotonic transformation.

Our major contributions are:

(1) **A New Paradigm for Layer Importance:** We introduce representational stability as a novel principle for assessing layer importance, shifting the focus from static parameter statistics to the dynamic robustness of a layer’s representations under controlled perturbations.

(2) **A Principled and Practical Framework:** We propose **PerturbPrune**, which operationalizes this principle by combining structured, PCA-guided perturbations with an information-theoretic metric, matrix difference entropy (DiME), to quantify layer stability while avoiding the biases of magnitude-based heuristics.

(3) **State-of-the-Art Performance with High Efficiency:** We demonstrate that **PerturbPrune** is a lightweight, plug-and-play module that consistently outperforms existing allocation methods across diverse LLMs and sparsity levels. It is particularly effective in high-sparsity regimes and incurs negligible computational overhead.

2 Methodology

In this section, we introduce **PerturbPrune**, a framework for non-uniform sparsity allocation in large language models, as illustrated in Figure 2.

2.1 Local Principal Subspace Approximation and Noise Generation

Given a pre-trained LLM and a calibration dataset $\mathcal{D}_{\text{cal}} = \{x_i\}_{i=1}^N$, we first perform a single forward pass to cache the clean hidden states $H^{(\ell, x_i)} \in \mathbb{R}^{T \times d}$ for each layer ℓ and input x_i , where T is the sequence length and d is the hidden dimension. For each layer ℓ , we estimate a local space from the empirical activation distribution. We center the token-level activations and compute the sam-

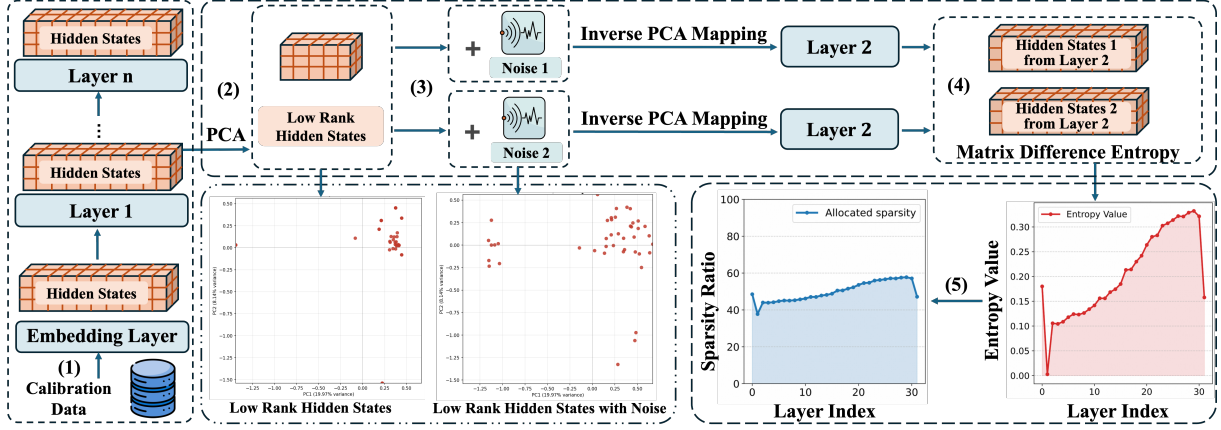


Figure 2: **Workflow of the PerturbPrune framework.** (1) Collect hidden states from each layer using calibration data; (2) Apply PCA for dimensionality reduction; (3) Generate two independent perturbations in the reduced space to measure layer stability; (4) Compute matrix difference entropy between outputs of perturbed versions; (5) High DiME scores (red curve) indicate stable layers assigned lower sparsity (blue curve), while low DiME scores reveal fragile layers that are pruned more aggressively. Stability scores are mapped to sparsity ratios via monotonic transformation.

ple covariance matrix over the T tokens, treating each token as a d -dimensional sample. We then perform PCA on this $d \times d$ covariance matrix, retaining the top- k principal components to obtain an orthonormal basis $V^{(\ell)} \in \mathbb{R}^{d \times k}$. To generate perturbations, we project the centered activations onto this low-dimensional space:

$$Z = (H^{(\ell, x_i)} - \mathbf{1}_T \mu^{(\ell)}) V^{(\ell)}, \quad (1)$$

where $Z \in \mathbb{R}^{T \times k}$ represents the coordinates in the space, $\mu^{(\ell)} \in \mathbb{R}^{1 \times d}$ is the mean activation, and $\mathbf{1}_T \in \mathbb{R}^{T \times 1}$ is a column vector of ones. We then add Gaussian noise in this space:

$$Z' = Z + \delta^{(\ell)}, \quad \delta^{(\ell)} \sim \mathcal{N}(0, \sigma^2 I_k), \quad (2)$$

where σ controls the noise intensity. The perturbed activations are mapped back to the original space via:

$$\tilde{H}^{(\ell, x_i)} = Z' (V^{(\ell)})^\top + \mathbf{1}_T \mu^{(\ell)}. \quad (3)$$

2.2 Layer Sensitivity Measurement

For each calibration sample x_i and each layer ℓ , we generate two independent perturbations of the hidden states, yielding $\tilde{H}_1^{(\ell-1)}$ and $\tilde{H}_2^{(\ell-1)}$. These are separately passed through the ℓ -th layer f_ℓ , producing outputs $O_1^{(\ell)}$ and $O_2^{(\ell)}$. We quantify the sensitivity of each layer using the DiME metric (Skean et al., 2023), which measures how much structural correspondence between paired inputs is preserved through the layer’s transformation. Specifically, DiME computes the difference in α -Rényi entropy

between aligned and permuted kernel matrices derived from the paired outputs. A higher DiME score indicates that the layer preserves strong alignment between its two perturbed outputs, implying robustness to input variations. Conversely, a lower DiME score suggests the layer disrupts input correspondence, revealing sensitivity to perturbations. Layers with lower DiME scores are thus assigned higher sparsity ratios, while layers with higher scores are preserved. Detailed formulas are provided in Appendix B.

2.3 Adaptive Layer-wise Sparsity from Robustness Scores

We translate robustness scores into non-uniform sparsity ratios. Layers with high S_ℓ (robust to perturbations) receive lower sparsity, while low S_ℓ (sensitive to perturbations) layers are pruned more aggressively. We first compute a relative sensitivity score C_ℓ for each layer:

$$C_\ell = S_{\max} - S_\ell, \quad S_{\max} = \max_\ell S_\ell. \quad (4)$$

After normalizing C_ℓ to $[0, 1]$, we apply a controlled mapping with maximum deviation hyperparameter λ :

$$a_\ell = 2\lambda \cdot \frac{C_\ell - C_{\min}}{C_{\max} - C_{\min}}, \quad \bar{a} = \frac{1}{L} \sum_{\ell=1}^L a_\ell. \quad (5)$$

The per-layer sparsity is then

$$\rho_\ell = \rho_{\text{global}} + a_\ell - \bar{a}, \quad (6)$$

Method	Strategy	Qwen3.5-9B		Llama 2-13B		Mistral-7B	
		PPL ↓	Acc ↑	PPL ↓	Acc ↑	PPL ↓	Acc ↑
Magnitude	Uniform	265547.16	34.87%	269.03	33.56%	22923.11	33.33%
	PerturbPrune	722.72 ↓99.7%	38.79% ↑11.2%	55.61 ↓79.3%	36.75% ↑9.5%	2444.03 ↓89.3%	35.63% ↑6.9%
Wanda	Uniform	93.82	39.48%	46.61	37.44%	60.34	36.85%
	PerturbPrune	71.61 ↓23.7%	42.52% ↑7.7%	17.49 ↓62.5%	45.94% ↑22.7%	29.13 ↓51.7%	40.64% ↑10.3%
SparseGPT	Uniform	100.25	38.26%	18.39	45.28%	23.12	44.02%
	PerturbPrune	75.79 ↓24.4%	40.12% ↑4.9%	12.79 ↓30.5%	48.96% ↑8.1%	17.28 ↓25.3%	45.43% ↑3.2%

Table 1: Performance comparison between uniform pruning and ours at 70% sparsity. We report PPL on WikiText2 and average zero-shot accuracy over seven tasks for three pruning methods. Blue ↓ and red ↑ denote PPL reduction and accuracy gain, respectively.

ensuring the global average sparsity equals the target ρ_{global} while allowing layer-wise deviations up to λ . In practice, we set λ to ensure all ρ_ℓ remain within $[0, 1]$. Finally, the computed $\{\rho_\ell\}$ is fed into any existing pruning algorithm.

3 Experiments

3.1 Experimental Settings

Models and Metrics. We evaluate PerturbPrune on the LLaMA-v1/v2/v3 (Touvron et al., 2023a,b), OPT (Zhang et al., 2022), and Qwen3.5 (Qwen Team, 2026)¹ model families, as well as Mistral-7B (Jiang et al., 2023) and Vicuna-7B (Zheng et al., 2023). Due to computational limits, we use models up to 35B parameters, in FP16, with max sequence length 2048. Most experiments run on a single NVIDIA RTX 4090 with a memory-expanded configuration (48GB), while Qwen3.5-9B and Qwen3.5-35B-A3B are evaluated on a single NVIDIA H20 (96GB).

Pruned models are assessed via PPL on WikiText-2 (Merity et al., 2016), where lower values indicate better performance, and via zero-shot tasks using the EleutherAI harness (Gao et al., 2024). The zero-shot benchmarks include RTE (Wang et al., 2018), WinoGrande (Sakaguchi et al., 2021), BoolQ (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), ARC-Easy/Challenge (Clark et al., 2018), and Hel-laSwag (Zellers et al., 2019), where higher accuracy indicates better performance.

Baselines. We compare PerturbPrune with four recent layer-wise sparsity allocation frameworks.²

¹For Qwen3.5, we apply pruning only to the text decoder component while keeping the vision encoder unchanged, as our focus is on language model compression.

²All baseline methods are evaluated using their official open-source implementations.

OWL (Yin et al., 2023) allocates pruning ratios based on activation outliers. AlphaPruning (Lu et al., 2024) leverages heavy-tailed self-regularization theory to guide layer-wise pruning decisions. MRP (Gao et al., 2025) uses singular value decomposition to measure layer redundancy for non-uniform pruning.

PerturbPrune is also integrated with three pruning backbones for fair comparison: magnitude pruning (Han et al., 2015), Wanda (Sun et al., 2023), and SparseGPT (Frantar and Alistarh, 2023).

Implementation Details. We use 128 calibration samples from C4 (Raffel et al., 2020) with sequence length 2048. All methods use identical calibration data, pruning backbones, and comparable hyperparameters, with a fixed random seed of 42. Unless otherwise specified, we retain the top- $k = 80$ principal components, which corresponds to approximately 11.76% of the token dimension, to construct local subspaces, and apply perturbations with a fixed noise scale $\sigma = 0.001$. Additional details are provided in Appendix B.

3.2 Main Results

Effect of Stability-Driven Sparsity Allocation under Extreme Sparsity. Table 1 compares uniform and stability-driven sparsity allocation at **70% sparsity**. Uniform allocation causes severe performance degradation across all pruning backbones and model families, with magnitude pruning collapsing most dramatically. In contrast, stability-driven non-uniform allocation substantially alleviates this issue, reducing PPL by up to one to two orders of magnitude and significantly improving zero-shot accuracy when combined with magnitude pruning. Moreover, consistent gains are also observed when pairing stability-driven allocation with

Method	LLaMA-7B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3.2-3B	OPT-6.7B	Mistral-7B	Vicuna-7B	Qwen3.5-9B
Dense	5.68	5.12	4.88	7.78	10.81	5.25	6.34	8.65
Magnitude	617.58	3679.78	11.22	14390.53	12807.26	48.46	685.13	33005.12
w. OWL	<u>104.63</u>	295.93	10.83	<u>11246.45</u>	17326.01	<u>23.12</u>	<u>133.54</u>	4530.50
w. Alpha	93.46	224.21	<u>9.67</u>	43836.41	17164.77	26.13	NaN	22326.37
w. MRP	119.42	<u>138.52</u>	12.19	15578.80	9675.96	42.73	89.80	<u>123.57</u>
w. Ours	156.70	135.78	9.49	7153.72	<u>15230.07</u>	21.47	89.80	51.20
Wanda	10.83	10.05	8.06	30.93	<u>15.57</u>	11.27	11.93	19.73
w. OWL	<u>9.57</u>	9.18	7.65	26.93	<u>16.23</u>	10.20	<u>11.40</u>	18.97
w. Alpha	9.79	9.81	<u>7.59</u>	30.29	27.92	10.78	12.02	20.01
w. MRP	11.25	<u>9.89</u>	8.62	26.36	36.71	10.46	13.37	22.93
w. Ours	9.32	8.63	7.21	<u>26.60</u>	15.40	<u>10.34</u>	10.52	20.45
SparseGPT	10.56	9.50	8.53	22.00	13.62	9.42	11.42	24.56
w. OWL	<u>9.54</u>	<u>9.10</u>	7.81	<u>20.43</u>	<u>13.80</u>	9.06	<u>11.10</u>	23.78
w. Alpha	10.01	9.71	<u>7.76</u>	23.82	20.24	9.46	11.82	15.74
w. MRP	10.73	9.78	8.39	21.19	17.47	9.43	12.68	36.00
w. Ours	9.51	8.61	7.18	19.94	13.62	<u>9.10</u>	9.97	24.40

Table 2: PPL results on WikiText2 at 60% sparsity. Alpha denotes AlphaPruning. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

stronger backbones such as Wanda and SparseGPT. These results indicate that stability-driven allocation is complementary to both simple and advanced pruning criteria, and becomes particularly critical in extreme sparsity regimes where uniform allocation is highly suboptimal.

Perplexity Evaluation. Table 2 reports PPL results on WikiText2 at 60% sparsity. Uniform magnitude pruning suffers severe performance collapse, showing that magnitude-based sparsity allocation is inadequate at high sparsity. Heuristic non-uniform strategies (e.g., OWL and AlphaPruning) yield only limited and unstable improvements. In contrast, stability-driven sparsity allocation consistently enhances performance across pruning backbones. Gains are most pronounced for magnitude-only methods, where stability-driven variants reduce PPL by orders of magnitude. When paired with activation-aware methods such as Wanda and SparseGPT, improvements are smaller but consistent, achieving the best or second-best PPL on most models. Overall, stability-driven allocation is robust across model families and particularly effective in high-sparsity regimes.

Zero-Shot Evaluation. Table 3 reports average accuracy over seven zero-shot tasks at 60% sparsity. Consistent with perplexity results, uniform sparsity causes severe degradation across all backbones, while non-uniform strategies consistently improve performance. Our stability-driven noise methods, frequently achieve the best or second-best

results among non-uniform variants under the same base pruner across model families. These methods are particularly effective with magnitude pruning, yielding large gains over heuristics such as OWL and AlphaPruning and in some cases matching or surpassing more advanced pruners. When combined with Wanda and SparseGPT, they provide stable improvements on LLaMA and OPT. Taken together, these findings demonstrate that stability-driven sparsity allocation effectively safeguards both language modeling quality and downstream semantic understanding in highly sparse regimes.

To confirm that our gains stem from principled importance estimation rather than arbitrary non-uniformity, we compare against 499 random configurations, ranking 1st out of 500 (Appendix E).

Effectiveness on Mixture-of-Experts Models

To validate the generalizability of PerturbPrune to different architectural paradigms, we further evaluate on Qwen3.5-35B-A3B, a state-of-the-art Mixture-of-Experts model. MoE architectures present unique challenges for pruning due to their sparse activation patterns and expert routing mechanisms. Table 4 presents comprehensive results across three pruning ratios (50%, 60%, and 70%), comparing PerturbPrune against uniform pruning and two recent advanced methods: Owl and AlphaPruning.

Our method demonstrates consistent superiority across all sparsity levels and competitive baselines. At 50% sparsity, PerturbPrune achieves the

Method	LLaMA-7B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3.2-3B	OPT-6.7B	Mistral-7B	Vicuna-7B	Qwen3.5-9B
Dense	60.08	59.69	63.05	57.24	32.57	64.37	60.36	67.29
Magnitude	36.12	39.53	43.87	32.52	32.57	45.08	45.21	33.52
w. OWL	40.93	43.41	47.59	<u>32.88</u>	<u>32.54</u>	49.56	47.56	35.75
w. Alpha	41.46	43.69	50.97	31.99	32.45	<u>48.86</u>	<u>48.17</u>	35.30
w. MRP	42.13	44.20	49.08	32.73	32.39	46.34	40.68	46.86
w. Ours	40.54	44.31	47.77	32.93	32.34	48.72	48.75	47.26
Wanda	50.60	50.19	55.46	41.76	46.11	51.09	51.90	55.89
w. OWL	<u>51.29</u>	<u>51.71</u>	<u>57.39</u>	<u>43.92</u>	<u>45.80</u>	<u>52.57</u>	53.03	<u>56.04</u>
w. Alpha	50.83	51.25	56.20	42.45	44.46	49.93	53.33	50.82
w. MRP	47.77	49.34	58.72	44.67	46.44	52.13	53.83	51.91
w. Ours	51.92	51.74	56.54	43.06	45.86	54.43	<u>53.74</u>	56.83
SparseGPT	51.51	51.52	55.94	45.85	<u>48.33</u>	54.44	52.72	49.94
w. OWL	51.61	<u>52.30</u>	56.90	47.15	47.89	54.77	53.88	50.19
w. Alpha	51.00	51.60	56.60	46.15	46.06	53.16	54.28	53.76
w. MRP	<u>52.06</u>	52.86	56.96	<u>47.09</u>	46.04	<u>55.12</u>	53.21	47.00
w. Ours	52.77	<u>52.30</u>	56.69	46.99	48.54	55.58	<u>53.95</u>	<u>51.27</u>

Table 3: Average Accuracy on seven zero-shot tasks at 60% sparsity. Alpha denotes AlphaPruning. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

Sparsity	Method	PPL↓	ARC-C↑	ARC-E↑	HellaSwag↑	RTE↑	WG↑	BoolQ↑	OBQA↑	Avg.↑
50%	Uniform	7.45	56.14	83.12	60.30	81.59	71.90	87.00	32.60	67.52
	Owl	7.25	<u>56.48</u>	83.29	60.74	82.67	<u>72.45</u>	<u>87.03</u>	32.80	67.92
	Alpha	7.11	57.25	83.12	60.39	<u>83.03</u>	72.06	87.34	32.20	67.91
	Ours	7.00	57.25	<u>83.25</u>	<u>60.59</u>	85.20	72.77	87.00	32.40	68.35
60%	Uniform	8.62	53.50	82.11	<u>58.52</u>	77.26	70.40	<u>84.56</u>	32.40	65.54
	Owl	8.38	55.63	82.66	58.79	<u>79.78</u>	69.69	86.73	31.20	<u>66.35</u>
	Alpha	8.15	55.97	84.34	58.00	<u>75.81</u>	71.67	81.80	31.80	65.63
	Ours	7.80	56.23	<u>83.63</u>	58.47	81.23	<u>71.11</u>	82.87	<u>32.00</u>	66.51
70%	Uniform	12.50	49.40	80.13	54.72	<u>71.48</u>	66.14	51.50	31.00	57.77
	Owl	<u>11.74</u>	49.74	80.35	<u>54.65</u>	74.01	<u>65.35</u>	70.40	30.20	60.67
	Alpha	12.17	49.23	80.81	52.70	67.51	62.67	54.40	<u>30.80</u>	56.87
	Ours	10.09	<u>49.57</u>	<u>80.72</u>	54.07	69.68	64.33	<u>62.20</u>	30.20	<u>58.68</u>

Table 4: Performance comparison on Qwen3.5-35B-A3B model with different pruning ratios using Wanda pruning method. Alpha denotes AlphaPruning. MRP is not included as it requires impractical computation time on this model size. PerturbPrune consistently outperforms uniform pruning across various sparsity levels. **Bold** values indicate the better result between methods.

best perplexity (7.00) and highest average accuracy (68.35), outperforming uniform pruning, Owl, and AlphaPruning. At 60% sparsity, the advantages become more pronounced with perplexity of 7.80 vs. 8.62 (uniform) and average accuracy of 66.51 vs. 65.54 (uniform), showing substantial gains on reasoning tasks like RTE (+3.97) and ARC-C (+2.73). At the aggressive 70% sparsity level, where uniform pruning exhibits severe degradation (perplexity: 12.50), PerturbPrune maintains significantly better perplexity (10.09, a 19.3% reduction) while preserving critical capabilities, particularly on BoolQ (+10.70 points).

These consistent improvements across multiple sparsity levels and against strong baselines validate

Method	Time	vs. PerturbPrune
PerturbPrune	7m 29s	1×
OWL	2m 21s	0.31×
MRP	2h 10m	17.4×
AlphaPruning	3h 1m	24.1×

Table 5: Wall-clock time for sparsity allocation on Qwen3.5-9B.

PerturbPrune’s effectiveness on MoE architectures, showcasing its ability to handle the unique sparse activation patterns of expert routing mechanisms.

3.3 Computational Efficiency

Table 5 compares the wall-clock time of different sparsity allocation methods on Qwen3.5-9B, where

Sparsity	50%	60%	70%	80%	90%
Speedup	1.19×	1.24×	1.30×	1.68×	3.09×

Table 6: Wall-clock inference speedup of unstructured sparse models over the dense model under the DeepSparse engine on an Intel Xeon Platinum 8469C CPU (benchmark numbers reported in prior work).

all results represent the computational cost of determining layer-wise sparsity ratios (excluding the actual pruning execution), averaged over 3 runs. PerturbPrune requires only 7 min 29 s, achieving $24\times$ and $17.4\times$ speedups over AlphaPruning and MRP, respectively, while maintaining superior quality as shown in Table 11. The efficiency stems from one-shot sensitivity estimation on cached hidden states, avoiding iterative convergence or expensive distribution computation required by competing methods. This makes PerturbPrune practical for large-scale deployment.

Beyond allocation-time efficiency, we further examine whether the resulting sparse models translate into end-to-end inference gains. Using the DeepSparse engine³ on an Intel Xeon Platinum 8469C CPU, unstructured sparse models achieve wall-clock speedups of $1.19\times$, $1.24\times$, $1.30\times$, $1.68\times$, and $3.09\times$ at sparsities of 50%, 60%, 70%, 80%, and 90%, respectively (Table 6), with near-linear scaling at moderate sparsity and significant acceleration at 90%. PerturbPrune’s non-uniform allocation is fully compatible with such sparsity-aware engines, which exploit per-layer sparsity regardless of whether the allocation is uniform or adaptive. We note that realizing wall-clock speedups from unstructured sparsity depends on specialized sparse kernels and hardware support; this is a shared challenge across the entire unstructured pruning field (SparseGPT, Wanda, OWL, and AlphaPruning all face the same issue), not specific to PerturbPrune, and our allocation strategy is orthogonal to the choice of inference backend.

3.4 Ablation Studies

Effect of Perturbation Scale. We analyze PerturbPrune’s sensitivity to the perturbation scale σ and find that matrix difference entropy is stable under moderate sparsity ($\leq 0.15\%$ variation) but more sensitive under aggressive pruning, with worst-case PPL degradation of about 1.8%. The optimal σ consistently lies in $[10^{-4}, 10^{-3}]$, which aligns well with our default principal subspace di-

³<https://github.com/neuralmagic/DeepSparse>

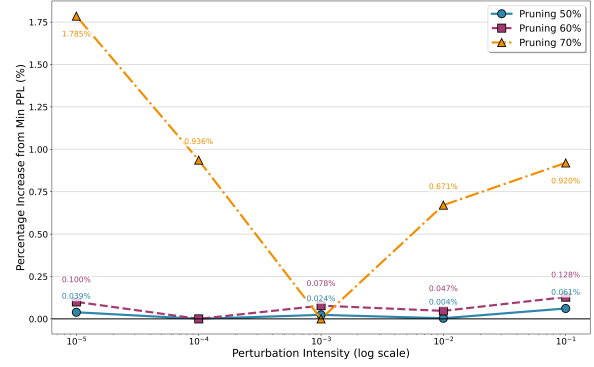


Figure 3: Relative Performance Degradation vs. Perturbation Intensity (σ).

mensionality, indicating a robust and effective operating point.

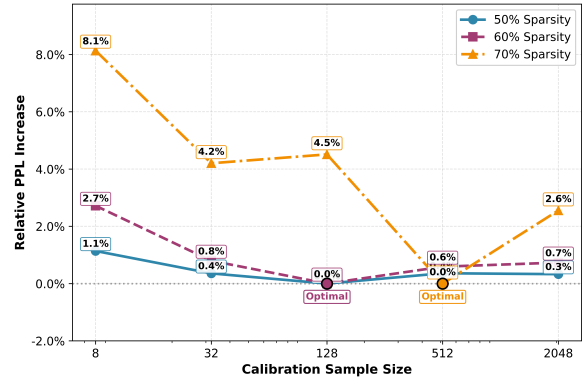


Figure 4: Relative PPL increase under different calibration data sizes at varying sparsity levels.

Impact of Calibration Data Size on PerturbPrune We analyze the effect of calibration data size on sparse models at 50%, 60%, and 70% sparsity. As shown in Figure 4, lower sparsity is largely insensitive to calibration size, while higher sparsity exhibits increased sensitivity but remains stable with moderate calibration sets (128–512 samples).

Effect of Principal Subspace Dimensionality.

We study the impact of space dimensionality by varying the retained principal component ratio. As shown in Figure 5, relative PPL remains stable across a broad range of component ratios at moderate sparsity levels (50% and 60%), confirming the robustness of our perturbation to dimensionality choice. At higher sparsity (70%), lower component ratios yield better performance, but PPL fluctuates and increases near full-rank, indicating that excessive feature directions introduce redundant noise and impair importance estimation. These re-

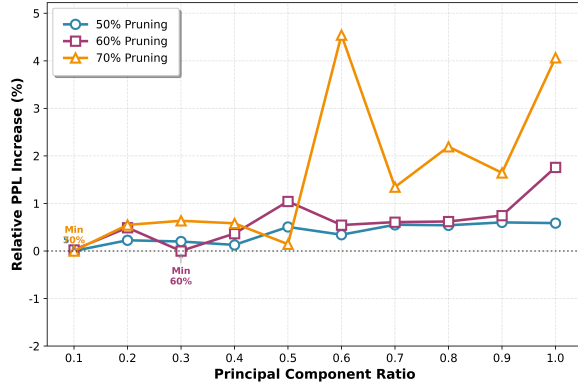


Figure 5: Relative PPL increase as a function of the retained principal component ratio when constructing principal spaces.

sults confirm the effectiveness of low-dimensional space projections and support using a relatively small component ratio to balance computational efficiency and estimation accuracy.

Effect of Matrix Difference Metrics. To validate the effectiveness of our proposed Difference Matrix Entropy (DiME) for measuring matrix dissimilarity, we conduct comprehensive ablation studies comparing DiME with three alternative metrics commonly used in the literature. Specifically, we compare against L2 Distance which measures the Frobenius norm of the difference between two matrices, Cosine Similarity which computes the cosine similarity between flattened matrix representations, and Activation Variance which measures the variance of activation differences across calibration samples.

Difference Metric	WikiText ↓	Avg. Acc. ↑
DiME (PerturbPrune)	8.63	51.74
L2 Distance	8.96	50.75
Cosine Similarity	11.57	48.45
Activation Variance	10.46	49.87

Table 7: Ablation study on different matrix difference metrics for computing DiME. We report perplexity on WikiText (lower is better) and average accuracy on downstream tasks (higher is better).

As shown in Table 7, our DiME metric achieves the best overall performance with an average accuracy of 51.74 across all downstream tasks, substantially outperforming L2 Distance (50.75), Activation Variance (49.87), and Cosine Similarity (48.45). DiME also demonstrates superior performance on language modeling, achieving a WikiText perplexity of 8.63, representing a 3.8% im-

provement over L2 Distance. The superior performance of DiME can be attributed to its ability to capture the information-theoretic divergence between weight matrices through entropy-based measurement. Unlike simple distances such as L2 norm or Cosine Similarity that only measure superficial matrix differences, DiME better reflects the semantic differences in model representations by quantifying the distribution of singular values in the difference matrix, providing a more principled measure of matrix dissimilarity that directly correlates with the preservation of model capabilities.

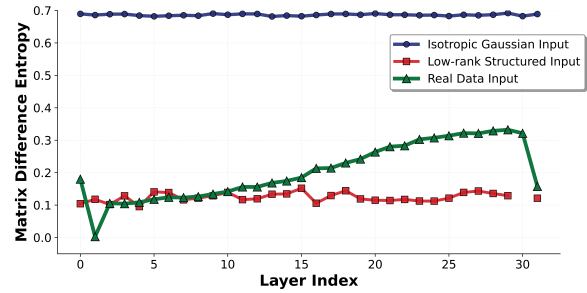


Figure 6: Layer-wise Matrix Difference Entropy under different input perturbations. Blue: perturbations on random high-dimensional hidden states (isotropic Gaussian noise). Red: perturbations on synthetic low-dimensional (3D) vectors prior to projection into high-dimensional space (low-rank structured noise). Green: real data. Notably, the red curve exhibits a dramatic outlier at layer 30, where the entropy sharply increases to 35.

3.5 Matrix Difference Entropy Analysis: Hidden States from Different Input Sources

A key assumption of PerturbPrune is that layer-wise stability measurements must be grounded in semantically meaningful inputs. To validate this assumption, we analyze the matrix difference entropy of hidden states under three input regimes with varying degrees of semantic content: Isotropic Gaussian Input (IGI), Real Data Input (RDI), and Low-rank Structured Input (LRSI), shown in Figure 6. IGI yields uniformly high entropy (~ 0.69) across all layers, providing no inter-layer differentiation for sparsity allocation. RDI exhibits a distinctive “check-mark” pattern: entropy drops sharply from ~ 0.18 (layer 0) to near 0 (layer 1), then rises to ~ 0.33 (layer 29), before decreasing in final layers. This variation reflects early denoising, progressive semantic disentanglement, and enables meaningful layer-wise sparsity differentiation. LRSI maintains flat low entropy (~ 0.12) through-

out, failing to provide discriminative signals despite its geometric structure.

Implications for method design. These results show that *semantic structure in inputs is essential* for stability-based allocation. Only RDI produces the structured entropy variation needed for non-uniform sparsity assignment. This justifies two design choices: (1) real calibration data is necessary—synthetic inputs yield uninformative metrics; (2) PCA-guided perturbations, derived from real activation statistics, probe semantically relevant directions, enabling DiME to produce discriminative layer-wise stability scores.

4 Conclusion

We propose **PerturbPrune**, a stability-driven framework for non-uniform sparsity allocation that probes layer robustness through structured perturbations and quantifies sensitivity via matrix difference entropy. Experiments across multiple architectures show PerturbPrune significantly outperforms existing methods in high-sparsity scenarios, preventing catastrophic degradation while maintaining computational efficiency. As a lightweight, plug-and-play module, PerturbPrune establishes representational stability as an effective proxy for layer importance, enabling efficient LLM deployment on resource-constrained platforms.

Limitations

Despite strong empirical results, PerturbPrune has several limitations. First, it introduces additional overhead during sparsity allocation, as principal subspace estimation, perturbations, and DiME computation require extra forward passes on calibration data. Although only 128 samples are used, the cost grows with model size and sequence length.

Second, hyperparameters such as the number of retained principal components ($\text{top-}k$), noise scale (σ), and maximum deviation (λ) may require tuning for new architectures or sparsity targets, despite working well across our tested configurations.

Third, perturbations are injected and evaluated per layer in isolation, so the stability estimate does not explicitly model cross-layer error propagation. We adopt this design because it guarantees linear-time allocation and allows each layer’s sensitivity to be cleanly attributed without confounded multi-layer interactions; empirically, per-layer stability already captures the dominant signal for allocation

across all tested configurations. Nonetheless, cascaded perturbation schemes that account for cross-layer effects remain a promising direction for future work.

Fourth, PerturbPrune exhibits architecture-dependent degradation in certain extreme configurations. On OPT-6.7B at 70% sparsity with Wanda (Table 16), the method produces elevated perplexity, while the same model with SparseGPT achieves the best result among all methods (PPL=21.07). This stems from OPT’s distinct architectural design (post-normalization with learned biases, different positional embeddings), which causes less consistent manifold geometry across layers and reduces the reliability of PCA-based space estimation at extreme sparsity. Future work could address this through architecture-adaptive stability estimation.

Finally, the method requires a minimum calibration set size (at least four samples) to ensure reliable DiME estimation, highlighting its dependence on data diversity even for small-scale calibration.

Ethical Statement

This work introduces PerturbPrune, a sparsity allocation framework for efficient compression of large language models (LLMs) through stability-driven pruning. We believe this research has positive societal impacts by reducing the computational and memory requirements of LLMs, thereby lowering energy consumption and carbon emissions associated with model deployment. This can democratize access to advanced AI technologies, enabling their use on resource-constrained devices and in underserved regions, while promoting sustainability in AI development.

However, we acknowledge potential risks. Pruned models may exhibit degraded performance on certain tasks or datasets, potentially amplifying existing biases in the original models or reducing robustness to adversarial inputs. To mitigate this, we evaluate PerturbPrune across diverse benchmarks, including zero-shot tasks, and recommend thorough downstream testing before deployment in real-world applications. All experiments use publicly available models (e.g., LLaMA, Qwen, OPT, Vicuna and Mistral) and datasets (e.g., C4, WikiText-2, EleutherAI harness), which do not involve sensitive personal data or human subjects, thus requiring no IRB approval. No private or proprietary data was used. Regarding misuse, while PerturbPrune is designed for benign effi-

ciency gains, compressed models could theoretically be used in harmful applications, such as generating misinformation. We encourage responsible use and advocate for safeguards like content moderation in deployed systems. Computationally, our experiments were conducted on a single NVIDIA RTX 4090 GPU and a single NVIDIA H20, minimizing environmental impact compared to training large models from scratch. Overall, we commit to ethical AI practices and believe PerturbPrune advances efficient, accessible AI without introducing novel ethical concerns beyond those inherent to LLMs.

Acknowledgments

This work was supported by the National Science Foundation of China (No. 62606048) and the Fundamental Research Funds for the Central Universities at Beijing Language and Culture University (21PT04).

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Saleh Ashkboos, Maximilian L Croci, Marcelo Genari do Nascimento, Torsten Hoefer, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.
- Ronen Basri and David Jacobs. 2016. Efficient representation of low-dimensional manifolds using deep networks. *arXiv preprint arXiv:1602.04723*.
- Shivansh Chhawri, Rahul Mahadik, and Suparna Roop. 2025. A systematic study of compression ordering for large language models. *arXiv preprint arXiv:2511.19495*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR.
- Chang Gao, Kang Zhao, Runqi Wang, Jianfei Chen, and Liping Jing. 2025. Maximum redundancy pruning: A principle-driven layerwise sparsity allocation for llms. *Preprint*, arXiv:2503.18377.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.
- Rafael Garcia, Tanja Munz, and Daniel Weiskopf. 2021. Visual analytics tool for the interpretation of hidden states in recurrent neural networks. *Visual Computing for Industry, Biomedicine, and Art*, 4(1):24.
- Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C Principe. 2014. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 61(1):535–548.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Tianteng Gu, Bei Liu, Bo Xiao, Ke Zeng, Jiacheng Liu, and Yanmin Qian. 2025. DenoiseRotator: Enhance pruning robustness for large language models via importance concentration. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.
- Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.
- Hanyu Hu, Pengxiang Zhao, Ping Li, Yi Zheng, Zhefeng Wang, and Xiaoming Yuan. 2025. Fasp: Fast and accurate structured pruning of large language models. *arXiv preprint arXiv:2501.09412*.
- Md Tauhidul Islam, Zixia Zhou, Hongyi Ren, Masoud Badii Khuzani, Daniel Kapp, James Zou, Lu Tian, Joseph C Liao, and Lei Xing. 2023. Revealing hidden patterns in deep neural network feature space continuum via manifold learning. *Nature Communications*, 14(1):8506.

- Albert Kjøller Jacobsen, Johanna Marie Gegenfurtner, and Georgios Arvanitidis. 2025. Staying on the manifold: Geometry-aware noise injection. *arXiv preprint arXiv:2509.20201*.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. *Mistral 7b*. Preprint, arXiv:2310.06825.
- Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. 2025a. T  yr-the-pruner: Unlocking accurate 50% structural pruning for llms via global sparsity distribution optimization. *arXiv preprint arXiv:2503.09657*.
- Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. 2024. Discovering sparsity allocation for layer-wise pruning of large language models. *Advances in Neural Information Processing Systems*, 37:141292  141317.
- Shusheng Li, Jiale Li, Yifei Qu, Xinwei Shi, Yanliang Guo, Ziyi He, Yubo Wang, and Wenjun Tan. 2025b. Semantic-eval: A semantic comprehension evaluation framework for large language models generation without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9675  9690.
- Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in neural information processing systems*, 37:9117  9152.
- Mingge Lu, Jingwei Sun, Junqing Lin, Zechu Zhou, and Guangzhong Sun. 2025. Lua-LLM: Learning unstructured-sparsity allocation for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702  21720.
- Jialin Mao, Itay Griniasty, Han Kheng Teoh, Rahul Ramesh, Rubing Yang, Mark K Transtrum, James P Sethna, and Pratik Chaudhari. 2024. The training process of many deep networks explores the same low-dimensional manifold. *Proceedings of the National Academy of Sciences*, 121(12):e2310002121.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. *Pointer sentinel mixture models*. Preprint, arXiv:1609.07843.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.
- Qwen Team. 2026. *Qwen3.5: Towards native multi-modal agents*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1  67.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99  106.
- Hang Shao, Bei Liu, and Yanmin Qian. 2024. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Oscar Skean, Jhoan Keider Hoyos Osorio, Austin J Brockmeier, and Luis Gonzalo Sanchez Giraldo. 2023. Dime: Maximizing mutual information by a difference of matrix-based entropies. *arXiv preprint arXiv:2301.08164*.
- Chuan Sun, Han Yu, Lizhen Cui, and Xiaoxiao Li. 2025. Efficient shapley value-based non-uniform pruning of large language models. *arXiv preprint arXiv:2505.01731*.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023a. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438  6447. PMLR.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for

natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. BESA: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, and 1 others. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. *Opt: Open pre-trained transformer language models*. Preprint, arXiv:2205.01068.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. *Judging llm-as-a-judge with mt-bench and chatbot arena*. Preprint, arXiv:2306.05685.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Related Work

Pruning Large Language Models. Large language models (LLMs) are highly over-parameterized, making pruning an effective way to reduce model size and inference cost (Zhu et al.,

2024; Chhawri et al., 2025). Pruning methods are typically unstructured, removing individual weights for high sparsity but irregular patterns (Frantar and Alistarh, 2023; Sun et al., 2023), or structured, removing neurons or attention heads for hardware speedup at the cost of accuracy (Xia et al., 2023; Ma et al., 2023; Ashkboos et al., 2024; Hu et al., 2025). Most rely on heuristic importance measures, which may overlook the geometric structure of hidden representations, motivating more structure-aware pruning criteria.

Uniform Sparsity Pruning. Early LLM pruning methods typically apply uniform sparsity ratios across layers, implicitly assuming homogeneous layer sensitivity. Representative approaches include SparseGPT (Frantar and Alistarh, 2023), which leverages approximate second-order information for one-shot unstructured pruning, Wanda (Sun et al., 2023), which uses activation-aware magnitude criteria without retraining, and LLM-Pruner (Ma et al., 2023), which performs structured pruning followed by lightweight recovery. While these methods achieve effective compression at moderate sparsity levels, enforcing uniform sparsity across layers often leads to suboptimal performance at higher pruning ratios due to neglecting layer-wise heterogeneity.

Non-Uniform Sparsity Allocation. Recognizing that LLM layers exhibit heterogeneous sensitivity and redundancy, recent work explores non-uniform, layer-wise sparsity allocation to better distribute pruning budgets. OWL (Yin et al., 2023) adjusts sparsity based on layer-wise activation outlier distributions, assigning lower sparsity to layers with more outlier features. AlphaPruning (Lu et al., 2024) leverages heavy-tailed self-regularization theory, using the power-law exponent of each layer’s weight spectral distribution as a layer importance indicator. SV-NUP (Sun et al., 2025) quantifies layer importance via Shapley values, while Týr-the-Pruner (Li et al., 2025a) employs global search to optimize sparsity allocation. MRP (Gao et al., 2025) uses singular value decomposition to quantify layer redundancy, iteratively updating redundancy estimates during the pruning process. Several recent works explore alternative routes to non-uniform allocation. Several recent works explore alternative routes to non-uniform allocation: BESA (Xu et al., 2024) learns per-layer pruning rates via blockwise reconstruction error minimization, PrunerGPT (Shao et al., 2024) maps per-layer

Algorithm 1 PerturbPrune

Input: Calibration set $\mathcal{D} = \{x_i\}_{i=1}^N$, global sparsity ρ_{global} , PCA dimension k , max deviation λ , perturbation strength σ

Output: Layer-wise sparsity allocation $\{\rho_\ell\}$

```
1: Phase 1: Sensitivity Estimation
2: for each layer  $\ell \in \{1, \dots, L\}$  do
3:   Initialize output collections:  $\mathcal{C}_1 = \emptyset, \mathcal{C}_2 = \emptyset$ 
4:   for each batch  $x_i \in \mathcal{D}$  do
5:      $H^{(\ell, x_i)} \leftarrow$  Extract input hidden states for batch  $x_i$   $\triangleright H^{(\ell, x_i)} \in \mathbb{R}^{T \times d}$ 
6:     Compute local basis  $V^{(\ell)} \in \mathbb{R}^{d \times k}$  via PCA on  $H^{(\ell, x_i)}$ , treating  $T$  tokens as samples in  $\mathbb{R}^d$ 
7:      $Z = (H^{(\ell, x_i)} - \mathbf{1}_T \mu^{(\ell)})V^{(\ell)}$   $\triangleright$  Project to local principal space
8:      $\tilde{H}_1^{(\ell, x_i)}, \tilde{H}_2^{(\ell, x_i)} \leftarrow \text{Perturb}(Z, \sigma, V^{(\ell, i)}, \mu^{(\ell)})$ 
9:      $\mathbf{O}_1^{(i)} = f_\ell(\tilde{H}_1^{(\ell, x_i)}), \quad \mathbf{O}_2^{(i)} = f_\ell(\tilde{H}_2^{(\ell, x_i)})$   $\triangleright$  Layer forward pass
10:     $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup \{\mathbf{O}_1^{(i)}\}, \quad \mathcal{C}_2 \leftarrow \mathcal{C}_2 \cup \{\mathbf{O}_2^{(i)}\}$   $\triangleright$  Collect outputs
11:   end for
12:   Concatenate all batch outputs:  $\mathbf{O}_1 = [\mathbf{O}_1^{(1)}, \dots, \mathbf{O}_1^{(N)}], \quad \mathbf{O}_2 = [\mathbf{O}_2^{(1)}, \dots, \mathbf{O}_2^{(N)}]$ 
13:   Compute layer sensitivity:  $S_\ell = \text{DiME}(\mathbf{O}_1, \mathbf{O}_2)$ 
14: end for
15: Compute layer contribution  $C_\ell := S_{\max} - S_\ell, \quad S_{\max} = \max_\ell S_\ell$ 
16: Normalize robustness score  $C_\ell$  across all layers  $a_\ell = 2\lambda \cdot \frac{C_\ell - \min(C)}{\max(C) - \min(C)}$ 
17: Phase 2: Allocate Non-Uniform Sparsity
18: Compute mean robustness score  $\bar{a} = \frac{1}{L} \sum_{\ell=1}^L a_\ell$   $\triangleright L$  is the total number of layers
19: Compute non-uniform sparsity for each layer  $\rho_\ell = \rho_{\text{global}} + a_\ell - \bar{a}$ 
20: return  $\{\rho_\ell\}$ 
```

sensitivity scores to sparsity ratios through a linear function, and Lua-LLM (Lu et al., 2025) optimizes the global mask end-to-end with learnable soft top- k thresholds, requiring gradient-based training. Orthogonal to allocation, DenoiseRotator (Gu et al., 2025) concentrates weight importance within each layer via learned orthogonal rotations, primarily enhancing uniform pruning and thus complementary to PerturbPrune. In Appendix E.5, we compare with these methods on Qwen3.5-9B and Qwen3.5-122B-A10B, and show that combining DenoiseRotator’s rotation with PerturbPrune’s non-uniform allocation yields further gains at high sparsity. Overall, adaptive sparsity allocation consistently outperforms uniform pruning, particularly under high compression ratios, by preferentially preserving critical layers.

Hidden State Manifolds and Manifold-Based Noise Injection. Prior studies show that hidden states in deep neural networks concentrate on low-dimensional manifolds and evolve along structured trajectories across layers, reflecting progressive representation refinement and disentanglement (Mao et al., 2024; Basri and Jacobs, 2016). This geomet-

ric structure implies that perturbations aligned with the underlying manifolds affect model behavior differently from isotropic noise in the ambient space. Motivated by this, manifold-aware operations on hidden representations, such as interpolation and topology-preserving analysis, have been shown to improve robustness and reveal layer-wise functional roles (Verma et al., 2019; Islam et al., 2023; Garcia et al., 2021). These findings suggest that perturbations provide a principled probe of layer importance. Accordingly, we estimate local principal spaces via PCA and inject structured noise in low-dimensional manifold coordinates to quantify pruning sensitivity using information-theoretic metrics.

B Detailed Implementation

Calibration Dataset We utilize a subset of the C4 dataset (Raffel et al., 2020) for sensitivity estimation, specifically the shard c4-train.00000-of-01024.json. Following the common practice in Sun et al. (2023), we randomly sample 128 sequences from this file, each truncated to a maximum length of 2,048 tokens, with a fixed random seed of 42 for reproducibility. These

samples are used to collect the hidden states and calculate the local sensitivity via PerturbPrune, as well as to compute neuron importance scores used in Wanda and SparseGPT.

Sensitivity Estimation (DiME) We quantify per-layer sensitivity using the *matrix-based difference of entropy* (DiME) (Giraldo et al., 2014). Given a layer ℓ , we collect two sets of output representations: $\mathbf{O}_1^{(\ell)}$ and $\mathbf{O}_2^{(\ell)}$, produced by passing two independent perturbations $\tilde{\mathbf{H}}_1^{(\ell-1)}$ and $\tilde{\mathbf{H}}_2^{(\ell-1)}$ through the ℓ -th layer f_ℓ . We compute the Gram matrices $\mathbf{K}_1 = \mathbf{O}_1^{(\ell)} \mathbf{O}_1^{(\ell)\top}$ and $\mathbf{K}_2 = \mathbf{O}_2^{(\ell)} \mathbf{O}_2^{(\ell)\top}$ using a *linear kernel*. This choice is motivated by the observation that semantic hidden states in LLMs are often linearly separable within their latent space (Hernandez et al., 2023), so a linear kernel effectively captures the principal structure of the representation manifold while ensuring computational efficiency.

The matrix-based Rényi’s α -order entropy of a normalized positive semi-definite matrix $\mathbf{A}$ (with $\text{tr}(\mathbf{A}) = 1$) is defined as:

$$S_\alpha(\mathbf{A}) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^r \lambda_i(\mathbf{A})^\alpha \right), \quad (7)$$

where $\{\lambda_i(\mathbf{A})\}$ are the eigenvalues of $\mathbf{A}$ and $r = \text{rank}(\mathbf{A})$. We set $\alpha = 1$ (the limiting case), which corresponds to the matrix analogue of von Neumann entropy (Giraldo et al., 2014):

$$S_1(\mathbf{A}) = - \sum_{i=1}^r \lambda_i(\mathbf{A}) \log \lambda_i(\mathbf{A}). \quad (8)$$

The DiME score for layer ℓ is then computed as:

$$S_\ell = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} S_\alpha \left(\frac{\mathbf{K}_1 \circ \pi \mathbf{K}_2 \pi^\top}{\text{tr}(\mathbf{K}_1 \circ \pi \mathbf{K}_2 \pi^\top)} \right) - S_\alpha \left(\frac{\mathbf{K}_1 \circ \mathbf{K}_2}{\text{tr}(\mathbf{K}_1 \circ \mathbf{K}_2)} \right), \quad (9)$$

where $\circ$ denotes the Hadamard (element-wise) product and Π is a set of random permutation matrices that break the sample correspondence between $\mathbf{K}_1$ and $\mathbf{K}_2$. A larger S_ℓ indicates that the true pairing has substantially lower entropy than random pairings, implying the layer’s output is highly aligned before and after perturbation (i.e., robust). A value of S_ℓ closer to zero suggests the perturbation disrupts the output structure (i.e., sensitive).

The resulting scores $\{S_\ell\}$ are then fed into the adaptive sparsity allocation described in Section 2.3, where robust layers (high S_ℓ) receive lower sparsity ratios to preserve their stable representations.

Interpretation of the DiME score. To make the allocation principle explicit, we interpret the two terms of S_ℓ in the equation above. The second term evaluates the entropy under the *true* pairing of the two perturbed outputs; it is low when the layer preserves the structural correspondence between the two perturbed inputs. The first term evaluates the same entropy under *random* pairings—the permutations $\pi \in \Pi$ destroy the sample correspondence—and thus serves as a correspondence-free baseline. Since S_ℓ is their difference (random-pairing entropy minus true-pairing entropy), it quantifies how much structural correspondence the layer preserves: a high S_ℓ means the layer *absorbs* perturbations—its outputs remain structurally aligned despite differently perturbed inputs—so it behaves as a compact, stable encoder whose parameters are efficiently utilized, and it is preserved; a low S_ℓ means the layer *amplifies* perturbations—the two outputs are no more aligned than random pairings—so its capacity is spent encoding irrelevant input variations, the hallmark of over-parameterization, and it tolerates more aggressive pruning. This is consistent with the representation-learning principle that good intermediate representations are invariant to irrelevant perturbations while preserving task-relevant information. Note that this “perturbation sensitivity” is orthogonal to conventional *pruning* sensitivity (quality degradation upon weight removal); the two lead to opposite allocation decisions, and conflating them makes our mapping appear counterintuitive. Empirically, inverting the mapping (pruning high- S_ℓ layers more) performs worse than uniform pruning for both DiME (Table 9) and an independent importance metric, OWL (Table 10), confirming the direction of the allocation.

Sparsity Allocation For the non-uniform sparsity allocation mechanism described in Section 2.2, the hyperparameter λ in Eq. 5 is set to 0.1 across all experiments. This value balances the layer-wise sensitivity with the global sparsity constraint, ensuring that the allocated pruning rates do not deviate excessively from the target budget while remaining adaptive to layer importance.

Pruning Backbones PerturbPrune is designed as a modular sparsity allocator that is compatible with various weight-level pruning methods. In our experiments, the actual pruning execution, including weight importance scoring, mask generation, strictly follows the official implementations of **Wanda** (Sun et al., 2023) and **SparseGPT** (Frantar and Alistarh, 2023). We do not modify the underlying weight-level pruning logic, ensuring that any performance gains are directly attributable to our stability-driven sparsity allocation.

C Sparsity Strategies Comparison

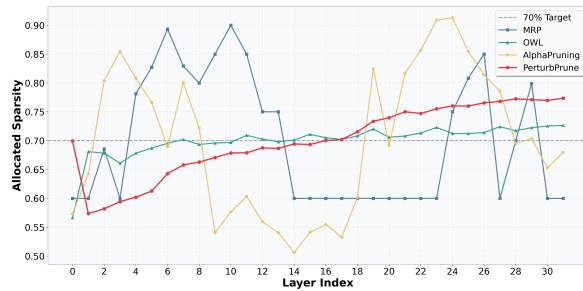


Figure 7: Layer-wise sparsity allocation strategies for Qwen3.5-9B at 70% overall sparsity. Our method achieves the best performance among all strategies.

The analysis reveals a strong correlation between the smoothness of layer-wise sparsity allocation and the model’s performance on downstream tasks. As shown in the figure, the sparsity allocation curves for MRP and AlphaPruning exhibit severe inter-layer volatility and irregularity. This unstable pattern likely disrupts the stable propagation of information through the network, leading to their relatively low average accuracies of 36.41% and 36.68% on zero-shot tasks, respectively.

In contrast, PerturbPrune produces a smooth, almost monotonically increasing sparsity schedule. This approach preserves more parameters in the shallower layers to process fundamental information and prunes more aggressively in deeper layers, which aligns with the intuition that deeper features have higher redundancy. This stable and well-structured allocation strategy enables PerturbPrune to achieve a state-of-the-art average accuracy of 42.52% on downstream tasks. These results demonstrate that incorporating stability into sparsity allocation guides the pruning process to create a more rational sparse structure, thereby preserving critical information while achieving efficient compression and ultimately improving the model’s generalization performance.

D Pseudo-code of PerturbPrune Algorithm

We provide the complete pseudo-code of our proposed PerturbPrune method in Algorithm 1.

E Additional Validation and Detailed Results

E.1 Effect of Allocation Strategy.

A key design choice in our method is the allocation strategy that determines how pruning ratios are distributed across layers based on their DiME scores. To validate our principle of "pruning less from stable layers," we compare three allocation strategies: (1) **Uniform** allocation that assigns equal pruning ratios to all layers regardless of their stability, (2) **Inverse Allocation** that adopts the opposite strategy by pruning more aggressively from stable layers (low DiME scores) and preserving unstable layers (high DiME scores), and (3) **PerturbPrune** that prunes less from stable layers and more from unstable layers based on the rationale that stable layers contain critical knowledge that should be preserved.

As shown in Table 9, our allocation strategy achieves the best performance with an average accuracy of 51.74 and a WikiText perplexity of 8.63, substantially outperforming both uniform allocation (50.19 accuracy, 10.05 PPL) and inverse allocation (50.42 accuracy, 10.59 PPL). Notably, the inverse allocation strategy, which aggressively prunes stable layers with high DiME scores, performs even worse than the uniform baseline. This significant performance gap validates our core hypothesis that layers with higher DiME scores encode more critical and robust knowledge that should be preserved during pruning.

E.2 Ablation Study on Inverse Sparsity Allocation

To validate the rationality of our proposed layer-wise sparsity allocation strategy, we conduct an inverse allocation experiment. Specifically, we reverse the sparsity distribution determined by OWL, assigning **higher sparsity** to layers identified as **more important** and **lower sparsity** to less critical layers. This counter-intuitive allocation serves as a strong ablation to verify whether our importance assessment correctly identifies critical layers.

Experimental Setup We evaluate three sparsity allocation strategies on Qwen3.5-9B at 70% overall

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
LLaMA2-7B (50%)	Dense	5.12	62.82	68.98	77.77	31.4	76.39	43.34	57.13	59.69
	Geodesic	<u>15.85</u>	60.65	<u>63.77</u>	63.73	26.6	<u>63.22</u>	<u>36.01</u>	50.21	52.03
	Brownian	13.60	61.01	64.56	64.07	27.6	65.28	35.41	49.86	52.54
	PerturbPrune	13.20	<u>58.12</u>	65.43	<u>63.03</u>	<u>29.2</u>	65.87	36.35	49.95	52.56
	Geodesic	<u>6.42</u>	53.43	67.24	72.60	<u>30.0</u>	73.06	39.08	53.08	55.50
	Brownian	6.43	53.79	68.19	74.65	30.0	71.59	39.33	52.89	55.78
	PerturbPrune	6.43	<u>54.15</u>	<u>67.48</u>	<u>74.62</u>	30.6	<u>72.10</u>	<u>39.25</u>	<u>52.86</u>	55.87
	Geodesic	6.47	<u>57.76</u>	<u>69.77</u>	76.21	29.2	71.42	<u>38.99</u>	52.66	<u>56.57</u>
	Brownian	6.50	56.68	70.01	76.27	<u>29.0</u>	71.76	38.23	52.63	56.37
	PerturbPrune	<u>6.49</u>	60.65	70.64	<u>76.09</u>	<u>29.0</u>	<u>71.72</u>	39.33	<u>52.54</u>	57.14

Table 8: Performance comparison on LLaMA2-7B at 50% sparsity in different noise perturbations. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column. Mag. is short for Magnitude and Sparse is short for SparseGPT.

Allocation Strategy	WikiText ↓	Avg. Acc. ↑
PerturbPrune	8.63	51.74
Uniform	10.05	50.19
Inverse Allocation	10.59	50.42

Table 9: Ablation study on different pruning allocation strategies. We report perplexity on WikiText (lower is better) and average accuracy on downstream tasks (higher is better).

sparsity: **Uniform** applies equal sparsity across all layers; **OWL** follows the importance-based allocation with lower sparsity for important layers; **OWL-Inv** reverses this allocation by assigning higher sparsity to important layers.

Results Table 10 summarizes the experimental results. The inverse allocation (OWL-Inv) leads to catastrophic performance degradation across all metrics. Perplexity increases dramatically from 78.77 to 145.40, indicating severe model collapse when critical layers are heavily pruned. The average accuracy on downstream tasks drops by 4.18 points compared to OWL (from 41.02 to 36.84), and even falls 2.64 points below the uniform baseline. This degradation is particularly severe on knowledge-intensive tasks, with ARC_E dropping 16.08 points and OBQA declining 2.80 points.

Analysis The experimental results validate several key aspects of our approach. First, the layer importance scores computed by our method successfully identify the truly critical layers in the network. Second, preserving more parameters in important layers (lower sparsity) is essential for maintaining model capability. Finally, the dramatic performance collapse under inverse allocation strongly confirms that our sparsity distribution methodology is well-grounded and effective.

E.3 Influence of Different Noise Perturbations

To investigate the impact of different geometric noise perturbations on pruning performance, we compare three distinct strategies for non-uniform sparsity allocation: principal, Geodesic, and Brownian, which are noise injection strategies proposed in Jacobsen et al. (2025). As shown in Table 8, we evaluate these strategies across three representative pruning methods—Magnitude, Wanda, and SparseGPT—on LLaMA2-7B at 50% sparsity.

Superior performance of our method. The proposed principal space perturbation consistently achieves the best or second-best performance across different pruning algorithms. With Magnitude pruning, it achieves the highest average accuracy of 52.56%, outperforming Geodesic (52.03%) and Brownian (52.54%). For SparseGPT, it attains the best average score of 57.14%, surpassing both alternatives (Geodesic: 56.57%, Brownian: 56.37%). It also demonstrates superior perplexity scores of 13.20 and 6.49 with Magnitude and SparseGPT respectively, indicating better language modeling capabilities.

Robustness across pruning methods. While the relative performance of different noise strategies varies slightly across pruning methods, our approach maintains consistent competitiveness, ranking first in 2 out of 3 methods by average accuracy. This robustness suggests that perturbation along principal activation directions provides a more stable and effective mechanism for exploring non-uniform sparsity patterns compared to alternative geometric approaches.

Pruning Rate	Method	PPL ↓	ARC-C	ARC-E	HellaSwag	RTE	WinoGrande	BoolQ	OBQA	AVG. ↑
70%	Uniform	93.82	20.31	43.31	30.77	52.71	52.25	62.39	14.60	39.48
	OWL	78.77	21.67	49.96	32.16	52.71	53.04	62.23	15.40	41.02
	OWL-Inv	145.40	17.32	33.88	28.55	53.43	49.96	62.11	12.60	36.84

Table 10: Performance comparison of different sparsity allocation strategies on Qwen3.5-9B at 70% pruning rate. OWL-Inv represents the inverse allocation where more important layers receive higher sparsity.

E.4 Detailed Quantitative Results

The following table presents the detailed performance of various models in terms of Perplexity (PPL) at sparsity levels of 50%, 60%, and 70%, as well as their specific results on 7 zero-shot tasks.

To summarize these results, we tally win/loss outcomes across all 72 model–pruner–sparsity configurations reported above. Among the four non-uniform allocation strategies (OWL, AlphaPruning, MRP, and PerturbPrune), PerturbPrune attains the lowest PPL in 46 configurations (63.9%), ranks second in 13, and ranks third or fourth in the remaining 13 (mean rank 1.67; no ties occur). Against uniform allocation, PerturbPrune wins in 59 of 72 configurations, and its win rate increases monotonically with sparsity, 66.7% at 50%, 87.5% at 60%, and 91.7% at 70%, confirming that stability-guided allocation is most beneficial when the pruning budget is tight.

The 26 losses concentrate in three regimes. First, on Mistral-7B, OWL wins 6 of 9 configurations, predominantly at 50–60% sparsity. Second, on Qwen3.5-9B pruned with SparseGPT, AlphaPruning is best at all three sparsity levels (e.g., 15.74 vs. our 24.40 at 60%). Third, at 70% sparsity with Magnitude pruning, all allocation methods degrade severely; four of the five losses there occur in configurations where every method reaches PPL values in the thousands, suggesting that rankings in this regime are dominated by the instability of the heavily pruned model rather than by allocation quality. Notably, in 18 of the 26 losses PerturbPrune still outperforms uniform allocation: even where it is not the best allocator, it rarely falls below the uniform baseline.

E.5 Comparison with Recent Allocation Methods

To situate PerturbPrune among recent non-uniform allocation methods, we additionally compare against BESA (Xu et al., 2024), Lua-LLM (Lu et al., 2025), PrunerGPT (Shao et al., 2024), and DenoiseRotator (Gu et al., 2025), using Wanda as

the pruning backbone and the same zero-shot evaluation protocol as in the main text.

Table 19 reports results on Qwen3.5-9B. PerturbPrune achieves the lowest PPL at all three sparsity levels and the highest average accuracy at 50% and 60%; at 70%, OWL obtains a slightly higher average accuracy while PerturbPrune still maintains the lowest PPL. Among the newly added baselines, BESA is competitive at 50% (the highest AVG of 68.57) but collapses catastrophically at 70% (PPL 94.40), as its blockwise reconstruction objective becomes ill-conditioned at extreme sparsity. Lua-LLM degrades steadily as sparsity increases (PPL 10.20 → 18.95 → 73.89) despite requiring gradient-based training, and PrunerGPT’s linear sensitivity-to-sparsity mapping fails even at moderate sparsity (PPL 14.65 at 50%) and collapses to near-random performance at 70% (AVG 38.52). This confirms that a linear mapping cannot capture the inter-layer dependencies that perturbation-based stability measurement detects.

Table 20 scales the comparison to Qwen3.5-122B-A10B. PerturbPrune achieves the highest average accuracy at 50% (69.48) and 70% (60.52); at 60%, BESA leads (65.22) with PerturbPrune close behind (64.68). PrunerGPT is competitive at 50% (AVG 68.67) but collapses at 70% (PPL 15.68, AVG 55.67)—the same “reasonable at moderate sparsity, catastrophic at high sparsity” pattern observed on the dense 9B model. Meanwhile, MRP’s combinatorial search becomes computationally prohibitive at this scale, whereas PerturbPrune completes sparsity allocation in minutes via linear-time PCA and DiME computation. Together with the Qwen3.5-35B-A3B results in the main text, these experiments establish coverage from 1.3B to 122B parameters.

Combination with DenoiseRotator. DenoiseRotator reshapes the within-layer weight landscape via learned orthogonal rotations, whereas PerturbPrune adapts the cross-layer pruning budget; the two approaches are therefore orthogonal. In Tables 21 and 22, we apply DenoiseRotator’s rotation

preprocessing and then compare uniform allocation with PerturbPrune’s non-uniform allocation, both using Wanda as the pruning backbone.

On the dense Qwen3.5-9B, non-uniform allocation consistently improves average zero-shot accuracy over uniform allocation after DenoiseRotator’s rotation, with gains increasing at higher sparsity (+0.57 / +2.15 / +2.56 AVG points at 50% / 60% / 70%); at 70%, it also substantially reduces PPL (84.70 $\rightarrow$ 67.84). This demonstrates that even after DenoiseRotator reshapes the weight distribution, non-uniform layer-wise allocation provides additional, complementary benefits. On the Qwen3.5-35B-A3B MoE model, the combination is most effective at 60% sparsity (PPL: 8.98 $\rightarrow$ 7.42; AVG: 64.68 $\rightarrow$ 67.15, +2.47 points), while at 50% and 70% results are mixed across individual tasks. We attribute this to the MoE architecture’s inherently sparse expert activation, which partially overlaps with PerturbPrune’s allocation mechanism, leaving less room for allocation-level gains at extreme sparsities. Overall, DenoiseRotator improves *what to prune* within each layer, while PerturbPrune improves *how much to prune per layer*, and their combination is particularly effective on dense models at high sparsity.

E.6 Hyperparameter Robustness

We further validate PerturbPrune’s insensitivity to hyperparameters at 60% sparsity. As shown in Table 23, varying the noise scale σ by $1000\times$ (0.001 $\rightarrow$ 1) changes PPL by only 0.02 and average accuracy by 0.01; doubling the PCA ratio from $0.1d$ to $0.2d$ changes PPL by merely 0.01. Together with a single $\lambda = 0.1$ used across all eight model families, these results confirm that practitioners can safely use the recommended defaults without architecture-specific tuning.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
Qwen3.5-9B (50%)	Dense	8.65	82.31	72.69	89.20	32.60	81.40	54.52	58.32	67.29
	uniform	1227.08	55.96	50.83	47.31	16.40	43.22	24.49	27.53	37.96
	OWL	227.43	53.79	52.57	55.96	19.80	59.93	<u>31.23</u>	31.38	43.52
	Alpha	345.60	52.71	52.96	53.88	16.60	50.21	27.56	28.47	40.34
	MRP	19.04	70.76	65.04	64.34	25.40	72.52	40.44	43.95	54.64
	PerturbPrune	18.90	74.73	<u>63.46</u>	66.67	26.00	73.15	40.44	44.87	55.62
	uniform	9.53	74.37	68.43	84.40	27.60	81.90	52.30	52.34	63.05
	OWL	10.76	78.70	69.85	84.89	27.40	80.26	<u>50.51</u>	51.20	63.26
	Alpha	<u>10.21</u>	<u>75.81</u>	<u>70.96</u>	82.84	<u>27.40</u>	79.84	48.63	49.97	62.21
	MRP	10.83	73.65	67.80	83.33	27.00	80.93	49.06	49.41	61.60
	PerturbPrune	11.57	73.29	71.03	81.47	26.60	79.88	50.17	51.15	61.94
	uniform	<u>11.91</u>	<u>75.09</u>	68.51	<u>77.37</u>	26.20	74.96	45.48	<u>50.28</u>	<u>59.70</u>
	OWL	12.14	66.43	68.59	74.95	29.00	76.14	47.35	<u>50.28</u>	58.96
	Alpha	10.00	67.87	70.96	80.92	<u>28.00</u>	80.64	50.51	51.62	61.50
	MRP	12.61	72.56	67.17	66.85	25.40	75.46	46.08	49.21	57.53
	PerturbPrune	13.41	77.98	67.40	71.41	27.40	<u>76.39</u>	47.10	50.18	59.69
	uniform	33005.12	51.62	50.12	39.08	16.20	29.21	22.78	25.64	33.52
	OWL	4530.50	51.26	50.51	46.15	16.40	36.57	22.61	26.73	35.75
	Alpha	22326.37	49.10	49.01	54.92	16.00	29.55	22.70	25.82	35.30
	MRP	<u>123.57</u>	61.73	57.46	65.23	18.80	56.90	31.06	36.82	46.86
	PerturbPrune	51.20	56.68	<u>56.91</u>	63.09	20.00	61.28	33.53	39.30	47.26
Qwen3.5-9B (60%)	uniform	19.73	67.51	64.80	77.77	24.00	73.91	39.51	43.70	55.89
	OWL	18.97	65.34	<u>64.88</u>	<u>78.23</u>	24.40	75.04	<u>40.61</u>	43.80	<u>56.04</u>
	Alpha	20.01	56.68	63.61	73.18	19.40	66.96	35.41	40.50	50.82
	MRP	22.93	62.09	62.98	71.38	21.00	69.87	36.52	39.55	51.91
	PerturbPrune	20.45	<u>65.70</u>	66.14	81.83	25.60	73.65	40.78	44.10	56.83
	uniform	24.56	56.68	61.88	<u>67.28</u>	20.40	66.58	35.49	41.27	49.94
	OWL	23.78	61.73	63.54	62.75	21.80	64.81	35.41	41.30	50.19
	Alpha	15.74	64.98	67.01	75.35	21.00	68.52	<u>35.75</u>	43.69	53.76
	MRP	36.00	56.32	60.30	66.39	18.00	60.61	30.46	36.87	47.00
	PerturbPrune	24.40	<u>62.09</u>	<u>64.72</u>	63.00	23.20	<u>66.62</u>	36.35	<u>42.91</u>	<u>51.27</u>
Qwen3.5-9B (70%)	uniform	265547.16	51.99	<u>52.09</u>	47.34	18.20	26.81	22.27	25.42	34.87
	OWL	104856.15	<u>55.60</u>	49.96	42.26	16.20	27.82	22.78	26.05	34.38
	Alpha	91953.74	49.10	50.59	42.14	15.80	26.09	22.10	25.85	33.10
	MRP	<u>1531.70</u>	57.04	52.17	62.17	<u>17.80</u>	39.69	24.57	30.47	40.56
	PerturbPrune	722.72	52.35	49.80	<u>61.59</u>	17.00	<u>38.80</u>	<u>22.87</u>	29.09	<u>38.79</u>
	uniform	93.82	52.71	52.25	<u>62.39</u>	14.60	43.31	20.31	30.77	39.48
	OWL	<u>78.77</u>	52.71	<u>53.04</u>	62.23	<u>15.40</u>	<u>49.96</u>	<u>21.67</u>	<u>32.16</u>	<u>41.02</u>
	Alpha	107.07	52.71	51.46	62.20	12.20	32.87	17.32	28.03	36.68
	MRP	92.46	52.71	47.91	60.55	11.20	36.99	17.24	28.26	36.41
	PerturbPrune	71.61	52.71	56.51	62.48	16.80	51.39	23.81	33.95	42.52
Qwen3.5-9B (70%)	uniform	100.25	53.07	53.83	<u>62.26</u>	12.40	36.91	19.80	29.53	38.26
	OWL	79.94	<u>52.71</u>	53.91	62.17	14.40	41.20	20.65	30.87	39.42
	Alpha	42.77	53.07	56.43	62.32	13.20	39.10	18.43	<u>31.41</u>	39.14
	MRP	109.37	53.07	50.59	61.04	12.60	33.25	17.83	28.20	36.65
	PerturbPrune	<u>75.79</u>	<u>52.71</u>	53.59	62.11	15.60	43.22	21.42	32.21	40.12

Table 11: Performance on Qwen3.5-9B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
LLaMA-7B (50%)	Dense	5.68	66.79	70.01	75.02	34.6	75.29	41.89	56.94	60.08
	Magnitude	uniform	17.73	54.51	59.43	54.53	22.6	58.80	33.53	46.98
		OWL	17.44	52.35	<u>63.61</u>	57.95	<u>26.6</u>	<u>61.07</u>	34.73	<u>49.12</u>
		AlphaPruning	<u>16.64</u>	<u>52.71</u>	55.72	<u>62.17</u>	16.2	43.52	20.56	40.36
		MRP	15.94	51.26	63.69	63.00	23.8	61.32	<u>34.13</u>	49.23
		PerturbPrune	21.34	<u>52.71</u>	60.93	59.17	27.4	59.68	33.87	46.21
	Wanda	uniform	7.35	55.23	65.90	71.07	28.4	69.19	<u>36.69</u>	51.90
		OWL	7.40	55.60	69.61	<u>72.84</u>	29.4	70.83	36.09	52.10
		AlphaPruning	<u>7.33</u>	<u>59.57</u>	67.64	<u>72.84</u>	30.0	69.78	36.43	51.77
		MRP	8.04	61.73	68.58	71.92	26.6	68.22	35.83	50.96
		PerturbPrune	6.44	54.15	67.17	76.02	30.0	72.56	38.48	53.20
	SparseGPT	uniform	7.29	56.32	68.19	72.91	28.0	<u>70.83</u>	35.58	51.32
		OWL	7.37	55.96	67.80	73.76	27.8	70.58	36.52	51.45
		AlphaPruning	7.37	55.23	68.75	74.01	27.6	70.45	<u>35.84</u>	51.21
		MRP	7.86	<u>59.20</u>	67.87	72.32	27.4	68.05	35.23	50.81
		PerturbPrune	<u>7.30</u>	61.01	<u>68.51</u>	<u>73.91</u>	27.4	71.09	36.52	51.84
LLaMA-7B (60%)	Magnitude	uniform	617.58	51.62	50.20	38.26	15.4	42.30	23.21	36.12
		OWL	<u>104.63</u>	53.07	54.85	<u>46.45</u>	21.2	47.10	26.45	37.38
		AlphaPruning	93.46	53.07	54.93	49.33	19.0	47.90	27.90	38.06
		MRP	119.42	53.07	57.22	41.47	23.4	48.27	29.43	42.00
		PerturbPrune	156.70	53.07	53.91	45.81	<u>21.2</u>	46.17	26.88	36.74
	Wanda	uniform	10.83	60.65	62.75	69.05	<u>25.2</u>	62.79	30.03	43.74
		OWL	<u>9.57</u>	58.12	<u>65.04</u>	69.76	24.8	62.96	31.57	46.79
		AlphaPruning	9.79	58.12	64.80	69.24	23.6	61.28	32.68	46.06
		MRP	11.25	61.37	65.58	67.34	23.6	58.79	32.00	45.24
		PerturbPrune	9.32	59.93	64.09	71.19	26.0	63.89	31.14	47.19
	SparseGPT	uniform	10.56	59.21	64.96	71.44	26.0	62.84	31.40	44.73
		OWL	9.54	55.96	65.98	71.71	25.4	63.47	32.59	46.18
		AlphaPruning	10.01	57.40	66.14	71.56	24.4	60.14	31.31	46.07
		MRP	10.73	63.18	66.46	72.54	22.8	60.86	32.68	45.92
		PerturbPrune	9.51	<u>62.82</u>	67.32	72.72	24.8	<u>62.96</u>	<u>32.34</u>	46.46
LLaMA-7B (70%)	Magnitude	uniform	50209.68	52.71	51.30	38.29	11.6	26.64	19.80	25.60
		OWL	22231.62	52.71	50.99	37.83	13.8	29.76	<u>22.27</u>	28.20
		AlphaPruning	1819.69	52.71	51.07	39.42	15.4	32.15	21.76	29.14
		MRP	30859.36	<u>52.70</u>	52.09	<u>38.38</u>	16.2	33.54	22.87	29.89
		PerturbPrune	<u>17742.13</u>	52.35	51.22	38.17	15.2	29.97	22.87	27.95
	Wanda	uniform	85.73	54.87	51.38	54.98	14.2	32.62	18.43	28.98
		OWL	22.27	55.96	60.06	<u>62.87</u>	<u>17.8</u>	48.06	25.17	<u>35.82</u>
		AlphaPruning	26.33	55.23	<u>60.85</u>	64.31	15.0	43.22	<u>25.26</u>	35.37
		MRP	25.03	62.09	63.14	62.20	15.8	43.69	26.88	36.44
		PerturbPrune	<u>23.07</u>	<u>56.68</u>	58.64	<u>62.87</u>	19.6	47.14	24.57	35.26
	SparseGPT	uniform	26.72	54.15	59.35	64.16	18.2	46.38	24.15	34.04
		OWL	<u>18.42</u>	53.07	62.51	66.27	20.8	48.36	26.62	37.38
		AlphaPruning	20.84	52.71	62.19	68.44	16.8	44.40	26.37	37.49
		MRP	21.81	52.70	64.40	65.41	18.8	44.49	27.90	37.61
		PerturbPrune	18.22	<u>53.43</u>	61.17	<u>67.55</u>	20.4	48.70	<u>27.13</u>	<u>37.51</u>

Table 12: Performance on LLaMA-7B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
LLaMA2-7B (50%)	Dense	5.12	62.82	68.98	77.77	31.4	76.39	43.34	57.13	59.69
	Magnitude	uniform	14.90	57.04	59.43	54.53	26.8	64.06	33.53	51.15
		OWL	19.12	55.60	63.06	60.92	29.4	64.18	34.47	50.93
		AlphaPruning	17.71	55.23	<u>64.33</u>	63.70	27.6	63.51	36.09	51.41
		MRP	16.55	58.12	62.51	58.56	29.4	66.37	37.37	53.91
		PerturbPrune	13.20	58.12	65.43	<u>63.03</u>	<u>29.2</u>	<u>65.87</u>	<u>36.35</u>	<u>49.95</u>
	Wanda	uniform	6.45	53.43	67.80	76.36	31.6	72.22	38.40	52.39
		OWL	6.89	53.79	67.80	75.44	30.2	71.97	38.82	53.03
		AlphaPruning	7.05	<u>54.87</u>	69.14	<u>75.60</u>	29.2	71.72	<u>39.16</u>	52.44
		MRP	7.50	62.82	69.06	70.86	30.0	71.68	38.57	56.52
		PerturbPrune	6.43	54.15	67.48	74.62	<u>30.6</u>	<u>72.10</u>	39.25	<u>52.86</u>
	SparseGPT	uniform	6.54	54.87	68.82	76.73	29.6	72.10	<u>39.25</u>	52.27
		OWL	6.96	55.96	69.85	75.75	29.4	71.93	39.08	52.95
		AlphaPruning	7.13	<u>61.73</u>	<u>68.98</u>	75.78	27.8	71.51	39.33	52.38
		MRP	7.57	64.62	68.98	76.18	28.4	70.20	38.05	<u>52.67</u>
		PerturbPrune	6.49	60.65	70.64	76.09	29.0	71.72	39.33	52.54
LLaMA2-7B (60%)	Magnitude	uniform	3679.78	50.90	53.12	47.77	20.0	44.82	25.09	39.53
		OWL	295.93	50.54	54.93	61.71	22.2	48.32	27.82	38.37
		AlphaPruning	224.21	50.90	55.88	58.62	24.4	49.75	<u>28.84</u>	37.41
		MRP	<u>138.52</u>	54.87	53.99	61.50	22.2	49.83	<u>28.33</u>	38.72
		PerturbPrune	135.78	<u>51.62</u>	<u>55.64</u>	59.79	<u>24.2</u>	49.66	30.03	39.26
	Wanda	uniform	10.05	53.43	65.19	67.25	26.4	64.77	30.72	43.59
		OWL	9.18	53.79	66.77	<u>67.09</u>	<u>28.0</u>	<u>67.72</u>	31.91	<u>46.67</u>
		AlphaPruning	9.81	53.79	68.11	66.67	25.4	65.28	<u>33.28</u>	46.19
		MRP	9.89	55.60	67.01	66.61	27.2	65.28	33.96	47.02
		PerturbPrune	8.63	<u>54.51</u>	<u>67.17</u>	64.68	28.2	67.76	<u>33.28</u>	46.56
	SparseGPT	uniform	9.50	53.43	65.82	72.66	26.2	<u>65.28</u>	31.91	45.31
		OWL	9.10	53.43	66.38	73.67	26.4	65.36	33.79	<u>47.09</u>
		AlphaPruning	9.71	54.51	<u>67.25</u>	73.09	24.4	64.48	30.97	46.49
		MRP	9.78	59.21	67.64	74.59	25.0	63.72	<u>32.59</u>	47.30
		PerturbPrune	8.61	<u>55.96</u>	<u>65.82</u>	73.61	26.8	65.07	31.40	46.57
LLaMA2-7B (70%)	Magnitude	uniform	52511.47	53.07	49.25	37.94	17.0	27.74	22.70	25.96
		OWL	61966.93	<u>52.35</u>	50.59	39.63	17.8	33.92	23.63	28.06
		AlphaPruning	NaN	53.07	49.72	45.17	19.2	34.22	24.15	28.99
		MRP	1356.14	<u>52.35</u>	53.67	49.08	17.2	38.26	24.40	32.82
		PerturbPrune	<u>17123.57</u>	53.07	<u>51.30</u>	38.47	<u>18.6</u>	31.94	<u>23.55</u>	<u>27.54</u>
	Wanda	uniform	72.90	<u>52.71</u>	51.07	40.64	12.0	30.98	18.77	27.85
		OWL	30.55	<u>52.71</u>	55.49	61.99	17.2	<u>43.14</u>	21.16	31.79
		AlphaPruning	31.76	<u>52.71</u>	<u>62.04</u>	<u>62.20</u>	<u>17.4</u>	43.01	<u>22.27</u>	<u>34.78</u>
		MRP	26.77	53.43	63.30	62.35	19.0	46.25	23.29	36.54
		PerturbPrune	<u>29.06</u>	<u>52.71</u>	54.78	62.17	16.2	<u>43.14</u>	20.99	31.45
	SparseGPT	uniform	23.69	53.07	57.85	65.38	18.6	44.15	21.59	33.31
		OWL	<u>19.72</u>	53.79	62.51	68.01	20.4	51.18	23.81	37.24
		AlphaPruning	21.06	57.04	<u>63.14</u>	65.75	19.2	47.05	23.63	36.94
		MRP	20.13	54.51	63.77	64.22	21.6	49.62	25.26	38.87
		PerturbPrune	18.44	<u>55.60</u>	60.14	<u>66.85</u>	19.4	<u>49.83</u>	<u>24.06</u>	36.23

Table 13: Performance on LLaMA2-7B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑	
LLaMA2-13B (50%)	Dense	4.88	65.34	72.22	80.61	35.2	79.46	48.46	60.04	63.05	
	Magnitude	uniform	6.49	55.96	65.35	57.65	27.6	70.54	38.48	54.39	52.85
		OWL	7.06	57.76	69.22	56.36	30.4	71.72	39.93	55.56	54.42
		AlphaPruning	6.55	62.45	69.06	70.92	31.8	74.96	42.32	56.25	58.25
		MRP	8.54	63.18	70.17	52.97	32.0	72.35	42.15	56.88	55.67
		PerturbPrune	6.14	55.60	69.77	58.56	31.6	72.10	40.27	55.40	54.76
	Wanda	uniform	5.69	59.93	70.80	81.28	32.0	75.72	43.09	57.06	59.98
		OWL	6.06	64.98	70.24	80.73	32.0	76.30	42.75	57.52	60.65
		AlphaPruning	5.99	58.84	71.43	80.03	33.0	76.73	43.09	57.00	60.02
		MRP	6.63	59.93	70.40	80.76	33.8	76.39	42.66	57.41	60.19
		PerturbPrune	5.57	65.70	70.80	80.34	31.2	76.73	42.92	56.71	60.63
	SparseGPT	uniform	5.73	66.79	71.90	81.71	32.8	75.08	41.13	56.12	60.79
		OWL	6.16	61.37	71.98	81.65	33.0	75.34	43.00	56.06	60.34
		AlphaPruning	6.07	64.62	72.14	81.44	31.6	75.00	41.98	55.88	60.38
		MRP	6.59	56.32	70.72	81.35	34.4	74.45	42.24	56.28	59.39
		PerturbPrune	5.63	62.09	72.77	81.59	33.2	76.47	42.49	55.59	60.60
LLaMA2-13B (60%)	Dense	11.22	52.71	57.30	47.40	21.8	56.19	27.05	44.66	43.87	
	Magnitude	OWL	10.83	53.07	63.46	49.14	23.6	62.12	32.76	49.00	47.59
		AlphaPruning	9.67	53.43	65.82	61.93	25.0	64.94	35.58	50.12	50.97
		MRP	12.19	56.68	65.67	45.78	25.6	62.58	35.49	51.73	49.08
		PerturbPrune	9.49	52.71	63.46	50.67	23.6	63.13	33.11	47.71	47.77
		Wanda	uniform	8.06	59.21	68.35	77.00	29.4	69.02	36.95	48.32
	OWL		7.65	60.65	69.30	79.82	29.2	72.18	38.40	52.18	57.39
	AlphaPruning		7.59	61.01	70.09	73.73	28.2	70.37	38.31	51.66	56.20
	MRP		8.62	66.06	69.77	81.19	30.0	72.14	39.16	52.71	58.72
	PerturbPrune		7.21	62.09	70.96	78.56	27.4	70.03	39.51	51.66	57.71
	SparseGPT	uniform	7.83	56.32	70.64	79.08	29.4	69.82	36.43	49.92	55.94
		OWL	7.81	59.21	71.03	81.44	28.8	69.91	36.86	51.05	56.90
		AlphaPruning	7.76	62.09	70.17	79.66	28.4	68.73	36.52	50.65	56.60
		MRP	8.39	57.04	71.35	82.02	30.4	69.70	36.86	51.38	56.96
		PerturbPrune	7.18	61.73	70.17	80.49	29.0	69.15	36.26	50.00	56.69
	LLaMA2-13B (70%)	Dense	269.03	52.71	49.41	38.69	14.6	31.31	20.65	27.53	33.56
Magnitude		OWL	54.59	52.71	56.67	38.96	15.6	40.36	24.32	35.41	37.72
		AlphaPruning	34.37	52.71	62.04	49.51	17.2	42.59	28.50	37.57	41.45
		MRP	33.89	52.71	58.17	42.45	18.2	45.29	27.47	41.44	40.82
		PerturbPrune	62.38	52.71	53.91	38.44	14.0	35.61	22.78	32.79	35.75
		Wanda	uniform	46.61	52.71	51.30	62.39	12.6	35.19	18.77	29.10
OWL			16.69	52.71	62.67	66.06	23.0	58.84	29.27	39.56	47.44
AlphaPruning			15.20	52.71	67.72	63.70	21.4	56.48	28.75	40.33	47.30
MRP			16.70	53.79	65.98	77.31	23.0	57.66	30.89	41.93	50.08
PerturbPrune			19.16	52.71	58.48	62.84	21.0	54.00	25.68	35.83	44.36
SparseGPT		uniform	18.39	53.07	61.09	66.85	20.4	53.20	25.85	36.53	45.28
		OWL	13.99	56.32	66.30	71.47	24.2	59.81	28.67	41.48	49.75
		AlphaPruning	13.49	53.79	68.43	71.01	22.8	57.28	29.18	41.23	49.10
		MRP	13.99	60.65	68.67	75.02	26.0	59.13	30.46	43.19	51.87
		PerturbPrune	13.55	54.51	64.64	68.87	23.6	56.48	27.22	38.62	47.71

Table 14: Performance on LLaMA2-13B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
LLaMA3.2-3B (50%)	Dense	7.78	54.51	69.69	73.27	31.2	74.54	42.15	55.31	57.24
	Magnitude	uniform	138.88	53.07	53.59	41.96	14.2	40.57	22.78	30.91
		OWL	96.52	51.99	54.14	<u>48.81</u>	<u>17.6</u>	41.25	23.21	35.26
		AlphaPruning	138.91	<u>52.71</u>	54.30	43.94	15.8	41.50	22.87	29.60
		MRP	<u>72.04</u>	47.65	56.27	49.63	<u>17.6</u>	41.75	25.77	35.57
		PerturbPrune	56.99	51.99	56.27	44.50	18.6	45.33	<u>25.09</u>	37.25
	Wanda	uniform	12.88	54.51	65.11	68.44	26.0	65.24	<u>32.51</u>	45.28
		OWL	12.98	54.87	65.43	70.00	24.0	65.82	32.42	46.16
		AlphaPruning	13.59	53.79	64.56	<u>70.64</u>	<u>25.0</u>	64.86	32.17	45.28
		MRP	14.52	54.87	65.51	71.59	22.2	63.85	33.79	46.34
		PerturbPrune	12.83	52.35	63.93	69.88	26.0	63.80	32.17	45.42
	SparseGPT	uniform	12.20	59.93	<u>67.17</u>	71.53	24.4	65.82	<u>34.64</u>	47.10
		OWL	<u>12.20</u>	61.01	66.46	<u>72.69</u>	27.6	64.94	33.96	47.85
		AlphaPruning	12.85	59.93	66.38	<u>72.35</u>	24.8	65.87	33.53	46.98
		MRP	13.43	<u>60.65</u>	65.82	72.81	<u>26.8</u>	<u>65.61</u>	34.98	48.35
		PerturbPrune	11.94	56.32	67.48	71.77	25.8	64.02	34.47	47.54
LLaMA3.2-3B (60%)	Magnitude	uniform	14390.53	50.18	51.70	38.13	<u>13.8</u>	26.52	20.82	26.50
		OWL	<u>11246.45</u>	53.79	<u>50.75</u>	38.01	<u>13.8</u>	26.52	<u>20.73</u>	26.53
		AlphaPruning	43836.41	51.26	49.72	38.07	13.2	25.55	19.80	26.33
		MRP	15578.80	51.99	48.70	<u>39.76</u>	14.8	27.15	20.14	<u>26.61</u>
		PerturbPrune	7153.72	<u>52.71</u>	47.83	41.83	<u>13.8</u>	27.90	19.62	26.82
	Wanda	uniform	30.93	<u>53.07</u>	54.54	64.01	16.0	48.53	22.10	34.06
		OWL	26.93	52.71	<u>58.56</u>	65.50	<u>17.2</u>	<u>51.30</u>	<u>25.51</u>	<u>36.68</u>
		AlphaPruning	30.29	52.35	57.38	63.00	16.2	49.71	23.38	35.15
		MRP	26.36	52.71	60.62	<u>64.50</u>	17.4	52.27	26.71	38.50
		PerturbPrune	<u>26.60</u>	52.71	56.59	63.15	17.4	50.80	25.09	35.67
	SparseGPT	uniform	22.00	<u>53.79</u>	61.48	68.04	19.0	54.17	25.77	38.70
		OWL	20.43	53.07	63.54	68.04	<u>21.2</u>	56.31	27.22	40.64
		AlphaPruning	23.82	54.51	62.12	69.69	18.0	54.04	26.28	38.38
		MRP	21.19	53.07	<u>63.61</u>	65.35	21.8	<u>56.06</u>	28.07	41.72
		PerturbPrune	19.94	53.07	63.85	69.08	20.2	55.47	27.22	40.01
LLaMA3.2-3B (70%)	Magnitude	uniform	102649.44	54.15	49.96	38.35	13.6	25.72	21.67	25.95
		OWL	85629.75	<u>52.35</u>	<u>49.41</u>	38.93	15.2	24.87	22.01	25.92
		AlphaPruning	126437.85	48.74	49.01	39.60	15.0	25.63	20.90	25.98
		MRP	102701.23	50.18	48.62	<u>39.48</u>	13.4	24.83	20.48	<u>26.02</u>
		PerturbPrune	<u>94965.54</u>	51.26	48.46	37.98	12.0	25.76	20.14	26.38
	Wanda	uniform	134.55	<u>52.71</u>	49.17	51.59	11.4	29.34	17.49	27.24
		OWL	122.42	<u>52.71</u>	49.33	<u>62.05</u>	<u>12.8</u>	32.87	18.26	<u>28.20</u>
		AlphaPruning	172.95	53.79	50.28	41.90	11.4	29.00	17.83	27.19
		MRP	101.20	<u>52.71</u>	49.33	62.17	13.0	34.39	18.86	29.07
		PerturbPrune	<u>108.91</u>	<u>52.71</u>	<u>50.20</u>	61.13	12.0	<u>32.95</u>	<u>18.60</u>	27.94
	SparseGPT	uniform	66.57	52.71	51.78	<u>62.14</u>	14.2	37.46	20.14	30.10
		OWL	53.61	52.71	53.59	63.49	13.4	38.68	20.14	32.43
		AlphaPruning	74.10	52.71	51.30	62.20	<u>13.6</u>	36.03	18.77	30.02
		MRP	<u>53.45</u>	52.71	55.72	<u>63.33</u>	13.8	41.08	21.67	33.35
		PerturbPrune	50.62	52.71	<u>55.56</u>	62.78	13.4	<u>40.19</u>	<u>20.31</u>	31.71

Table 15: Performance on LLaMA3.2-3B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑	
OPT-6.7B (50%)	Dense	10.81	55.23	65.43	66.12	27.6	65.66	30.55	50.47	51.58	
	Magnitude	uniform	1026.47	52.71	50.75	38.04	17.8	39.81	20.22	29.36	35.53
		OWL	<u>526.11</u>	52.71	53.12	<u>38.41</u>	19.2	<u>48.99</u>	<u>23.04</u>	37.34	<u>38.97</u>
		AlphaPruning	763.50	52.71	53.51	38.04	21.4	45.54	24.49	<u>36.62</u>	38.90
		MRP	609.99	51.99	54.78	41.50	<u>20.0</u>	49.45	21.76	35.32	39.26
		PerturbPrune	486.36	52.71	53.28	37.95	18.4	45.79	21.42	35.31	37.84
	Wanda	uniform	12.20	52.71	61.72	65.23	25.6	63.51	28.41	46.85	49.15
		OWL	12.63	52.71	60.77	63.67	24.0	62.58	27.39	46.07	48.17
		AlphaPruning	14.23	53.79	61.40	63.33	24.6	59.39	27.05	45.23	47.83
		MRP	17.84	<u>53.43</u>	60.06	62.20	23.4	56.23	26.88	42.86	46.44
		PerturbPrune	12.09	52.71	61.88	<u>63.98</u>	27.0	62.96	27.47	<u>46.77</u>	<u>48.97</u>
	SparseGPT	uniform	11.79	53.43	63.85	66.48	25.8	63.22	28.84	47.65	<u>49.90</u>
		OWL	12.18	52.71	64.72	66.18	<u>25.6</u>	<u>63.05</u>	<u>28.16</u>	<u>46.93</u>	49.62
		AlphaPruning	13.18	54.15	63.54	64.95	<u>25.6</u>	61.07	26.71	46.39	48.92
		MRP	13.89	52.35	63.54	62.91	24.8	60.14	26.96	45.42	48.02
		PerturbPrune	11.73	<u>53.07</u>	<u>64.40</u>	65.99	27.4	63.09	28.24	47.55	49.96
OPT-6.7B (60%)	Dense	12807.26	52.71	<u>50.51</u>	37.83	12.6	26.43	21.76	26.16	32.57	
	Magnitude	OWL	17326.01	53.07	49.64	37.83	12.8	28.16	19.97	26.30	32.54
		AlphaPruning	17164.77	<u>52.71</u>	49.33	37.83	12.4	27.69	<u>20.82</u>	<u>26.38</u>	<u>32.45</u>
		MRP	9675.96	51.99	51.30	37.86	10.6	27.44	21.16	26.35	32.39
		PerturbPrune	<u>12286.28</u>	<u>52.71</u>	47.51	37.83	14.0	27.40	21.50	26.41	32.29
		Wanda	uniform	15.41	52.71	59.19	62.26	22.8	57.74	25.26	42.80
	OWL		16.23	52.71	58.72	62.51	22.8	57.58	<u>24.49</u>	41.82	<u>45.80</u>
	AlphaPruning		27.92	55.60	57.46	62.17	19.2	51.81	25.26	39.73	44.46
	MRP		36.71	<u>53.43</u>	56.51	62.17	18.2	48.32	24.40	37.64	42.95
	PerturbPrune		<u>15.57</u>	52.71	59.75	<u>62.32</u>	<u>22.2</u>	<u>57.28</u>	24.32	<u>42.54</u>	45.87
	SparseGPT	uniform	13.62	53.07	63.46	64.25	23.6	61.83	27.13	44.94	48.33
		OWL	<u>14.50</u>	51.99	64.01	<u>63.64</u>	<u>24.6</u>	59.93	<u>27.05</u>	44.00	47.89
		AlphaPruning	20.24	<u>54.51</u>	61.17	62.17	22.6	54.46	25.85	41.67	46.06
		MRP	17.47	53.43	60.22	62.23	22.0	57.11	25.60	41.66	46.04
		PerturbPrune	13.80	54.87	62.75	63.15	26.0	60.90	27.56	<u>44.64</u>	48.54
	OPT-6.7B (70%)	Dense	17361.36	52.71	49.96	37.83	12.6	25.59	19.80	25.98	32.07
Magnitude		OWL	17428.32	52.71	49.49	37.83	13.2	24.79	21.59	25.88	32.21
		AlphaPruning	23876.17	52.71	50.83	37.83	12.2	25.13	20.31	25.63	32.09
		MRP	13398.36	52.71	<u>50.75</u>	37.83	<u>12.8</u>	<u>25.72</u>	<u>20.90</u>	<u>26.11</u>	32.40
		PerturbPrune	<u>15230.07</u>	52.71	50.83	37.83	<u>12.8</u>	25.93	20.73	25.88	<u>32.39</u>
		Wanda	uniform	160.00	53.07	51.38	60.83	11.2	35.73	17.92	26.92
OWL			44.16	<u>52.35</u>	51.54	<u>62.08</u>	15.0	44.32	19.11	31.01	<u>39.34</u>
AlphaPruning			<u>98.08</u>	50.18	51.85	62.17	<u>15.4</u>	<u>40.74</u>	<u>22.35</u>	33.03	39.39
MRP			180.46	51.99	<u>51.78</u>	62.17	14.0	39.02	24.06	<u>32.06</u>	39.29
PerturbPrune			1035.47	50.54	48.54	38.99	13.4	31.73	17.24	26.70	32.45
SparseGPT		uniform	20.78	52.35	<u>57.77</u>	<u>62.20</u>	20.6	54.92	24.06	<u>38.72</u>	<u>44.37</u>
		OWL	22.59	54.51	56.91	62.14	18.8	52.53	<u>24.15</u>	37.35	43.77
		AlphaPruning	76.33	46.57	57.38	62.17	<u>19.6</u>	42.26	25.00	35.43	41.20
		MRP	35.47	<u>53.79</u>	55.41	62.17	19.0	48.91	23.04	35.83	42.59
		PerturbPrune	<u>21.19</u>	52.71	60.14	62.26	20.4	53.96	23.63	38.86	44.57

Table 16: Performance on OPT-6.7B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
Mistral-7B (50%)	Dense	5.25	67.87	73.95	83.61	32.6	80.89	50.43	61.21	64.37
	Magnitude	uniform	7.92	55.96	66.46	71.04	27.2	72.60	41.30	55.89
		OWL	8.04	59.57	<u>69.53</u>	69.11	<u>27.8</u>	72.31	40.70	<u>56.89</u>
		AlphaPruning	8.50	57.76	68.11	<u>70.95</u>	26.6	73.74	<u>42.06</u>	56.51
		MRP	8.38	58.84	69.61	72.75	27.4	73.36	42.32	56.35
		PerturbPrune	8.47	58.48	68.82	67.61	29.2	72.18	<u>42.06</u>	57.08
	Wanda	uniform	6.55	57.76	71.27	82.84	27.4	74.66	<u>42.75</u>	55.17
		OWL	6.65	58.84	70.48	82.66	26.8	74.49	41.64	55.01
		AlphaPruning	<u>6.69</u>	<u>59.93</u>	69.46	81.47	26.4	74.71	39.68	53.62
		MRP	7.02	59.57	70.64	<u>82.26</u>	26.6	74.79	41.72	54.76
		PerturbPrune	8.10	61.37	66.93	81.35	28.0	76.43	46.33	51.03
	SparseGPT	uniform	6.47	66.06	71.03	82.75	<u>27.2</u>	75.72	42.83	56.18
		OWL	6.61	65.34	71.90	82.78	26.4	75.38	42.58	55.85
		AlphaPruning	6.62	63.90	<u>71.74</u>	82.69	26.6	74.83	41.47	54.89
		MRP	6.96	<u>64.26</u>	72.53	82.23	27.0	<u>75.59</u>	<u>43.86</u>	60.12
		PerturbPrune	8.05	63.54	68.11	81.56	28.0	76.05	46.33	51.68
Mistral-7B (60%)	Magnitude	uniform	48.46	52.71	57.93	43.09	21.2	63.05	31.48	45.08
		OWL	<u>23.12</u>	54.87	63.38	<u>56.45</u>	22.4	65.32	35.41	<u>49.08</u>
		AlphaPruning	26.13	53.43	<u>62.27</u>	64.56	21.4	61.70	31.91	46.78
		MRP	42.73	57.40	60.62	51.01	20.0	61.32	29.69	44.64
		PerturbPrune	21.47	55.23	62.43	53.52	<u>21.8</u>	<u>63.43</u>	<u>35.07</u>	49.55
	Wanda	uniform	11.27	54.87	66.22	71.16	21.8	67.09	32.00	44.47
		OWL	10.20	57.76	<u>67.09</u>	<u>74.07</u>	<u>22.8</u>	67.68	32.76	45.81
		AlphaPruning	10.78	57.40	66.30	65.14	21.2	65.36	30.03	44.07
		MRP	10.46	57.40	67.96	71.87	21.8	66.25	<u>33.53</u>	46.09
		PerturbPrune	<u>10.34</u>	55.23	64.40	81.10	24.8	71.97	37.54	45.96
	SparseGPT	uniform	9.42	58.84	68.03	76.97	23.8	69.15	35.24	49.08
		OWL	9.06	60.29	68.59	76.42	24.0	<u>69.40</u>	35.32	49.38
		AlphaPruning	9.46	56.68	<u>67.72</u>	77.22	22.8	67.80	32.68	47.25
		MRP	9.43	<u>61.01</u>	69.38	76.27	<u>24.8</u>	69.07	<u>36.26</u>	49.49
		PerturbPrune	9.10	62.09	68.35	<u>76.82</u>	25.2	69.91	37.20	49.47
Mistral-7B (70%)	Magnitude	uniform	22923.11	52.71	52.33	37.77	13.0	30.13	20.48	26.89
		OWL	833.18	53.43	<u>51.54</u>	<u>38.20</u>	<u>16.6</u>	35.56	21.50	30.60
		AlphaPruning	1861.42	52.35	52.25	40.43	15.4	34.68	<u>22.18</u>	29.81
		MRP	15367.86	52.71	48.54	38.59	12.8	30.22	19.45	26.91
		PerturbPrune	2526.09	53.43	51.46	37.89	16.8	35.90	22.27	32.04
	Wanda	uniform	60.34	52.71	51.14	59.82	13.0	34.55	17.92	28.81
		OWL	38.17	52.71	54.62	62.17	14.6	40.19	18.94	31.53
		AlphaPruning	39.54	52.71	55.49	<u>62.51</u>	13.2	38.59	18.77	31.43
		MRP	35.04	52.71	58.96	62.42	15.0	41.25	20.65	34.24
		PerturbPrune	19.73	52.71	59.27	62.60	16.6	57.45	26.37	36.58
	SparseGPT	uniform	23.12	<u>54.15</u>	59.19	66.30	17.6	51.73	22.78	36.40
		OWL	18.54	54.51	62.59	63.70	20.0	54.71	25.17	37.77
		AlphaPruning	20.58	52.71	62.51	62.35	16.8	49.20	22.61	35.69
		MRP	18.96	54.51	<u>64.33</u>	63.36	19.0	55.22	27.30	38.98
		PerturbPrune	17.28	<u>54.15</u>	65.19	62.39	20.2	52.95	<u>27.05</u>	<u>38.84</u>

Table 17: Performance on Mistral-7B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

	Method	PPL ↓	RTE ↑	WinoGrande ↑	BoolQ ↑	OBQA ↑	ARC-E ↑	ARC-C ↑	HellaSwag ↑	avg. ↑
Vicuna-7B (50%)	Dense	6.34	63.90	69.46	80.92	33.0	75.63	43.17	56.47	60.36
	Magnitude	uniform	<u>15.36</u>	61.01	64.72	75.26	26.6	65.61	38.40	54.67
		OWL	19.78	62.09	65.43	<u>75.66</u>	27.8	<u>66.37</u>	<u>38.14</u>	55.25
		AlphaPruning	18.80	<u>63.18</u>	<u>65.75</u>	78.56	<u>29.2</u>	65.11	35.92	<u>55.48</u>
		MRP	28.98	65.34	59.59	68.87	25.6	62.33	34.98	52.35
		PerturbPrune	15.15	65.34	66.30	75.35	29.6	66.54	37.97	51.39
	Wanda	uniform	7.94	56.32	68.03	<u>80.21</u>	29.8	71.59	39.93	53.14
		OWL	8.65	54.51	67.56	79.91	29.6	72.05	39.93	53.02
		AlphaPruning	8.72	<u>59.57</u>	<u>67.96</u>	80.92	<u>29.8</u>	71.13	39.51	52.64
		MRP	9.38	66.06	66.69	76.73	27.0	68.86	36.01	51.77
		PerturbPrune	7.96	55.23	67.17	79.91	30.2	<u>71.55</u>	<u>40.27</u>	52.92
	SparseGPT	uniform	7.91	54.51	66.93	80.89	28.8	70.37	39.76	52.90
		OWL	8.48	54.51	66.69	79.97	30.6	71.42	39.59	53.10
		AlphaPruning	8.63	<u>65.34</u>	69.14	80.24	30.0	<u>71.42</u>	39.42	53.10
		MRP	9.19	67.51	66.85	77.80	26.4	69.57	37.54	52.29
		PerturbPrune	7.85	56.32	<u>67.72</u>	80.46	29.0	71.51	40.53	53.21
Vicuna-7B (60%)	Magnitude	uniform	685.13	55.23	56.35	63.06	21.6	51.05	30.38	45.21
		OWL	<u>133.54</u>	54.15	59.75	66.85	<u>24.8</u>	54.50	30.12	42.72
		AlphaPruning	NaN	58.12	60.62	68.04	23.0	53.71	<u>31.57</u>	42.11
		MRP	149.29	53.07	56.91	40.86	20.8	45.33	26.79	40.98
		PerturbPrune	89.80	58.84	<u>59.67</u>	68.93	25.0	<u>54.29</u>	31.66	42.83
	Wanda	uniform	11.93	52.71	64.80	72.63	27.2	66.58	33.70	45.66
		OWL	<u>11.40</u>	53.43	65.11	73.88	<u>27.4</u>	67.59	<u>35.07</u>	47.89
		AlphaPruning	12.02	<u>58.48</u>	66.77	73.58	27.2	65.11	<u>35.07</u>	47.03
		MRP	13.37	68.59	64.17	76.27	24.6	61.36	<u>35.07</u>	46.74
		PerturbPrune	10.52	55.96	66.77	<u>75.11</u>	27.8	<u>67.55</u>	35.41	47.60
	SparseGPT	uniform	11.42	53.07	65.67	77.80	24.8	65.32	35.15	47.22
		OWL	<u>11.10</u>	53.79	66.22	78.81	26.6	67.68	35.58	48.49
		AlphaPruning	11.82	57.76	67.25	78.93	27.6	66.08	34.39	47.96
		MRP	12.68	64.62	64.48	76.30	24.2	61.99	33.70	47.15
		PerturbPrune	9.97	52.71	<u>65.43</u>	<u>78.90</u>	<u>27.6</u>	<u>67.42</u>	36.52	48.48
Vicuna-7B (70%)	Magnitude	uniform	NaN	<u>53.07</u>	48.46	38.23	15.4	30.09	23.46	33.57
		OWL	<u>18423.73</u>	52.71	<u>50.75</u>	<u>50.21</u>	16.4	33.75	24.66	28.79
		AlphaPruning	NaN	52.71	48.61	58.29	18.4	36.45	25.00	29.66
		MRP	27806.41	53.79	51.30	38.26	<u>17.4</u>	30.35	22.35	28.42
		PerturbPrune	7733.47	52.35	50.12	47.52	16.0	<u>35.23</u>	<u>24.49</u>	28.20
	Wanda	uniform	59.47	52.71	50.75	62.14	14.0	32.58	17.75	28.80
		OWL	<u>31.62</u>	52.71	58.72	64.07	20.4	52.27	24.57	35.83
		AlphaPruning	37.62	52.71	59.19	62.42	17.8	48.65	26.02	36.50
		MRP	38.71	66.79	62.19	63.76	17.8	44.15	26.19	37.85
		PerturbPrune	29.62	<u>56.32</u>	56.67	63.36	<u>19.2</u>	53.03	24.57	<u>35.95</u>
	SparseGPT	uniform	30.48	53.43	57.70	66.30	17.0	47.52	25.51	34.88
		OWL	24.58	52.71	61.72	70.64	20.8	54.25	29.27	39.14
		AlphaPruning	26.85	<u>54.51</u>	<u>61.80</u>	70.21	20.0	52.36	26.45	39.25
		MRP	25.83	55.96	<u>61.80</u>	68.10	19.2	46.55	28.50	<u>39.05</u>
		PerturbPrune	21.40	53.43	60.54	72.72	21.0	<u>56.52</u>	<u>28.07</u>	38.63

Table 18: Performance on Vicuna-7B under different sparsity levels. **Bold** and underlined values indicate the best and second-best results, respectively, among different non-uniform sparsity allocation strategies for the same pruning method within each column.

Sparsity	Method	PPL ↓	ARC-C	ARC-E	HellaSwag	RTE	WinoGrande	BoolQ	OBQA	AVG ↑
50%	Uniform	7.45	56.14	83.12	60.30	81.59	71.90	87.00	<u>32.60</u>	67.52
	OWL	7.25	56.48	<u>83.29</u>	60.74	82.67	72.45	87.03	32.80	67.92
	AlphaPruning	<u>7.11</u>	57.25	83.12	60.39	83.03	72.06	<u>87.34</u>	32.20	67.91
	BESA	7.57	56.48	84.22	58.70	86.28	74.03	89.30	31.00	68.57
	Lua-LLM	10.20	50.94	81.36	52.03	80.87	69.14	85.78	27.20	63.90
	PrunerGPT	14.65	51.02	80.43	50.55	66.43	65.90	85.05	28.40	61.11
	Ours	7.00	57.25	83.25	<u>60.59</u>	<u>85.20</u>	<u>72.77</u>	87.00	32.40	<u>68.35</u>
60%	Uniform	8.62	53.50	82.11	<u>58.52</u>	77.26	70.40	84.56	32.40	65.54
	OWL	8.38	55.63	82.66	58.79	<u>79.78</u>	69.69	<u>86.73</u>	31.20	<u>66.35</u>
	AlphaPruning	<u>8.15</u>	<u>55.97</u>	84.34	58.00	75.81	71.67	81.80	31.80	65.63
	BESA	9.93	52.13	82.91	52.99	78.70	69.38	88.50	29.60	64.89
	Lua-LLM	18.95	40.18	73.94	43.93	72.20	64.95	76.63	24.80	56.66
	PrunerGPT	16.48	40.70	72.85	42.59	57.76	60.46	71.99	23.40	52.82
	Ours	7.80	56.23	<u>83.63</u>	58.47	81.23	<u>71.11</u>	82.87	<u>32.00</u>	66.51
70%	Uniform	12.50	49.40	80.13	54.72	<u>71.48</u>	66.14	51.50	31.00	57.77
	OWL	<u>11.74</u>	49.74	80.35	<u>54.65</u>	74.01	<u>65.35</u>	70.40	30.20	60.67
	AlphaPruning	<u>12.17</u>	49.23	80.81	<u>52.70</u>	67.51	<u>62.67</u>	54.40	<u>30.80</u>	56.87
	BESA	94.40	20.56	43.35	30.75	52.71	51.30	<u>62.57</u>	14.60	39.41
	Lua-LLM	73.89	21.41	47.85	31.45	52.70	52.95	62.53	15.40	40.61
	PrunerGPT	51.31	27.90	45.08	30.27	35.64	50.79	61.29	18.70	38.52
	Ours	10.09	<u>49.57</u>	<u>80.72</u>	54.07	69.68	64.33	62.20	30.20	<u>58.68</u>

Table 19: Comparison with recent sparsity allocation methods on Qwen3.5-9B, using Wanda as the pruning backbone. **Bold** and underlined values indicate the best and second-best results among all methods in each column for each sparsity level. Lua-LLM requires approximately one hour of gradient-based training on 4×A100 GPUs; all other methods are training-free.

Sparsity	Method	PPL ↓	ARC-C	ARC-E	HellaSwag	RTE	WinoGrande	BoolQ	OBQA	AVG ↑
50%	Uniform	6.84	59.13	83.16	<u>64.67</u>	75.81	74.82	85.02	36.00	68.37
	OWL	<u>6.48</u>	59.47	83.12	64.37	72.20	74.35	84.77	35.40	67.67
	AlphaPruning	6.20	60.15	83.21	64.39	78.34	73.88	78.44	35.20	67.66
	BESA	7.16	<u>61.09</u>	84.55	61.58	79.06	75.14	75.75	35.60	67.54
	PrunerGPT	7.11	59.78	82.70	62.65	<u>78.75</u>	<u>75.02</u>	<u>85.61</u>	<u>36.20</u>	<u>68.67</u>
	Ours	6.83	61.43	<u>83.71</u>	64.77	78.23	74.74	86.91	36.60	69.48
60%	Uniform	8.13	54.86	80.47	<u>62.26</u>	65.34	71.43	81.77	33.80	64.28
	OWL	7.80	56.74	80.81	62.43	66.79	71.03	79.11	35.40	64.62
	AlphaPruning	7.45	55.80	<u>83.00</u>	62.16	65.04	<u>72.93</u>	74.80	<u>35.20</u>	64.13
	BESA	8.16	58.19	84.51	58.37	<u>66.23</u>	74.82	80.65	33.80	65.22
	PrunerGPT	<u>7.64</u>	56.91	81.99	59.97	65.81	72.85	<u>81.22</u>	32.60	64.48
	Ours	8.20	<u>57.94</u>	82.70	61.45	64.62	72.38	78.69	35.00	<u>64.68</u>
70%	Uniform	10.66	46.08	72.05	<u>57.70</u>	68.23	63.93	78.50	31.60	<u>59.73</u>
	OWL	<u>10.79</u>	46.84	72.69	56.94	66.79	63.93	<u>79.14</u>	30.80	59.59
	AlphaPruning	12.58	45.05	76.64	54.07	61.01	63.93	<u>77.34</u>	32.00	58.58
	BESA	11.23	47.69	<u>75.16</u>	51.96	68.70	<u>64.35</u>	72.60	30.60	58.72
	PrunerGPT	15.68	39.76	72.39	47.21	61.37	64.17	78.78	26.20	55.67
	Ours	10.80	<u>47.65</u>	73.02	57.98	<u>68.51</u>	65.35	79.36	<u>31.80</u>	60.52

Table 20: Comparison on Qwen3.5-122B-A10B with the Wanda backbone. **Bold** and underlined values indicate the best and second-best results among all methods in each column for each sparsity level. Lua-LLM is omitted as it runs out of memory on the smaller Qwen3.5-35B-A3B; MRP could not complete its search within a practical time budget on this 122B model.

Sparsity	Method	PPL ↓	ARC-C	ARC-E	HellaSwag	RTE	WinoGrande	BoolQ	OBQA	AVG ↑
50%	DR + Uniform	8.95	52.86	82.36	53.48	75.74	68.85	85.08	27.80	63.74
	DR + PerturbPrune	10.23	51.38	83.94	53.82	76.13	70.56	85.72	28.60	64.31
60%	DR + Uniform	18.31	40.27	75.23	44.30	69.59	64.77	79.24	24.80	55.70
	DR + PerturbPrune	19.29	41.51	75.26	44.89	68.03	66.36	83.29	25.60	57.85
70%	DR + Uniform	84.70	22.67	45.18	33.52	53.17	53.18	63.60	18.60	41.42
	DR + PerturbPrune	67.84	24.13	52.68	34.19	54.63	58.64	64.38	19.20	43.98

Table 21: DenoiseRotator (DR) rotation preprocessing followed by uniform vs. PerturbPrune allocation on Qwen3.5-9B (Wanda backbone). **Bold**: better of the two allocations in each column for each sparsity level.

Sparsity	Method	PPL ↓	ARC-C	ARC-E	HellaSwag	RTE	WinoGrande	BoolQ	OBQA	AVG ↑
50%	DR + Uniform	7.06	58.25	84.46	60.99	81.53	75.79	88.73	34.60	69.19
	DR + PerturbPrune	7.13	58.69	83.96	62.56	71.76	76.14	89.18	33.80	68.01
60%	DR + Uniform	8.98	55.20	81.84	60.47	81.99	57.33	83.70	32.20	64.68
	DR + PerturbPrune	7.42	55.17	82.72	61.42	83.87	68.59	84.51	33.80	67.15
70%	DR + Uniform	10.45	52.07	80.56	54.75	80.13	69.26	68.37	32.00	62.45
	DR + PerturbPrune	10.31	52.69	79.52	55.36	72.53	65.89	70.52	30.60	61.02

Table 22: Same setting as Table 21, on Qwen3.5-35B-A3B.

Hyperparameter	Setting	PPL ↓	Avg. Acc. ↑
Noise scale σ	0.001	11.11	61.99
	1	11.13	61.98
PCA ratio	0.1 <i>d</i>	11.11	61.99
	0.2 <i>d</i>	11.12	62.00

Table 23: Hyperparameter robustness at 60% sparsity. PerturbPrune is highly insensitive to both the noise scale and the PCA ratio.