Generation-Drift-Guided Block Pruning for Large Language Models

Zhiqi Huang **Heedong Kim** **Lintong Zhang** **Seong-Whan Lee***
Department of Artificial Intelligence
Korea University, Seoul, South Korea
{zhiqi_huang, hd_kim, zhanglintong, sw.lee}@korea.ac.kr

Abstract

Pruning reduces the model size and inference cost of large language models (LLMs), yet preserving performance across diverse downstream tasks remains challenging. Existing structured pruning methods commonly rely on local discrepancy measures or teacher-forced predictive criteria, such as Block Influence (BI) and perplexity (PPL). While these methods are effective on classification benchmarks, the resulting pruned models often degrade severely on generative reasoning tasks. We study redundancy in Transformer-based models and observe that pruning-induced generation drift is strongly correlated with the functional importance of modules, suggesting a more faithful criterion for identifying redundant modules. Based on this insight, we propose GDPruner, a calibration-data-free, generation-drift-guided block pruning method. GDPruner constructs lightweight self-generated probe trajectories, estimates module importance using tail-position generation drift, and removes redundant modules through adaptive search. Extensive experiments show that GDPruner achieves the best overall performance compared with existing structured pruning baselines in most settings, with particularly clear gains on complex generative reasoning tasks, offering a promising direction for generation-aware LLM pruning.

1 Introduction

Large language models (LLMs) have achieved remarkable success across a wide range of natural language understanding and generation tasks [1, 2]. This success is largely driven by recent advances in model scaling, pretraining, and instruction tuning [3–6]. However, these gains come with substantial computational and deployment costs, making efficient inference increasingly important [7]. This need has motivated broad efforts on efficient LLMs, including pruning [8–11], quantization [12–17], and distillation [18, 19]. Among these directions, pruning is particularly attractive because it can directly reduce model size and inference cost while preserving much of the original performance [20, 21]. In particular, structured pruning removes entire groups of parameters or computational units, making it especially promising for practical acceleration of LLMs [22–28]. This raises the question of how to identify which structures can be removed with minimal damage to model performance.

Layer pruning [29–31] is a widely used structured pruning strategy, but its coarse granularity has motivated recent work on finer-grained block pruning [32, 33]. Many existing structured pruning methods estimate redundancy using either Block Influence (BI)-based local discrepancy signals [29, 34] or perplexity (PPL)-based teacher-forced evaluation [31, 32]. While these criteria often preserve reasonable performance on classification benchmarks, they can lead to substantial degradation on complex generative reasoning tasks. As illustrated in Fig. 1, such degradation can manifest as

*Corresponding author.

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

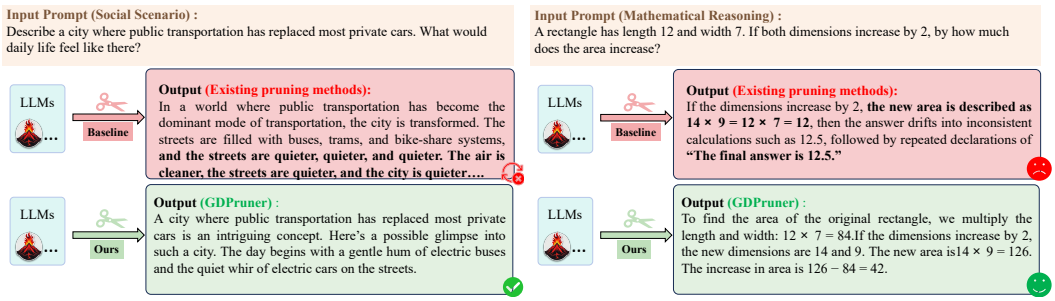


Figure 1: Comparison of generated outputs between existing pruning methods and GDPruner.

repetitive outputs, broken semantic progression, inconsistent intermediate reasoning, and ultimately incorrect final answers, limiting the practical deployability of pruned LLMs.

To understand why pruning affects classification and generative reasoning differently, we compare their computational characteristics. Classification often makes a final decision through a single open-loop forward pass, making it less exposed to recursive error accumulation. In contrast, generative reasoning relies on multi-step autoregressive decoding, where each predicted token conditions subsequent hidden states and future predictions [35]. As a result, pruning-induced perturbations can be fed back into later decoding steps and accumulate into substantial trajectory deviation [36]. This suggests that module redundancy is not strictly task-invariant: modules that appear removable under short-horizon classification may still be important for stable autoregressive generation. Conversely, our analysis in Sec. 2.2 shows that modules inducing little generation drift are strong candidates for broad redundancy, as their removal does not materially alter the dense model's generation dynamics. Appendix A provides an intuitive illustration of this error-accumulation process.

This insight leads us to revisit the notion of redundancy in LLM pruning. Existing redundancy estimates rely on local representational changes or teacher-forced predictive quality, making them more closely aligned with open-loop prediction than with closed-loop autoregressive generation. As a result, the modules they identify as redundant may correspond only to a task-biased notion of redundancy: such modules may be removable for classification, yet remain essential for maintaining coherent multi-step reasoning. To overcome this limitation, we propose **GDPruner**, a calibration-data-free, generation-drift-guided block pruning approach that evaluates module importance using generation drift at tail decoding positions. Extensive experiments demonstrate that GDPruner better preserves complex generative reasoning while achieving stronger overall performance than existing structured pruning baselines in most settings. Our key contributions are summarized as follows:

- We conduct an empirical analysis of pruning across different task types and show that generation drift provides a more reliable signal for identifying broadly redundant modules, offering new insights into structured pruning for LLMs.
- We propose GDPruner, a calibration-data-free, generation-drift-guided block pruning method that redefines module importance through its impact on generation trajectory stability, shifting pruning from preserving local computations to preserving generative reasoning dynamics.
- Extensive experiments demonstrate that GDPruner achieves stronger overall performance than existing structured pruning baselines in most settings, with particularly clear gains on complex generative reasoning tasks.

2 Generation Drift as a More Faithful Criterion for Pruning

2.1 Preliminaries

We consider an LLM with L Transformer layers. Following the standard pre-normalization architecture [1, 37], each layer consists of two sequential residual blocks: a multi-head self-attention (MHA) block and a multi-layer perceptron (MLP) block. In this work, we treat each MHA block and each MLP block as an individual pruning unit. Let H_{i-1} denote the hidden states entering layer i, and let $\mathrm{LN}(\cdot)$ denote the layer normalization operation. The forward computation of layer i is given by

$$H'_i = H_{i-1} + \mathrm{MHA}_i\big(\mathrm{LN}(H_{i-1})\big), \tag{1}$$

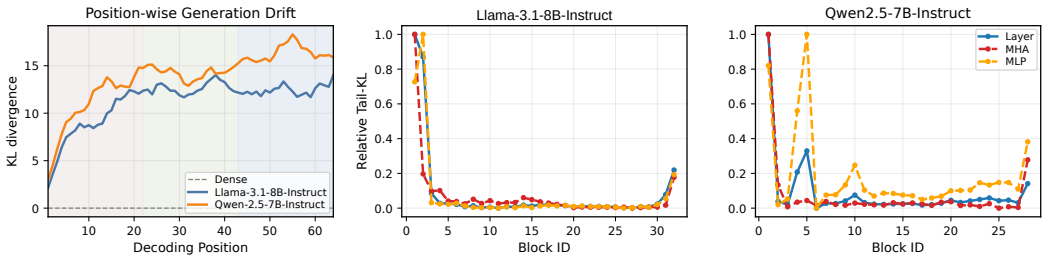


Figure 2: Empirical analysis of pruning-induced generation drift: generation drift becomes more pronounced at later positions (Left), while different layers and modules exhibit substantially different sensitivities (Middle and Right). Such fine-grained heterogeneity motivates block-level pruning.

$$H_i = H'_i + \mathrm{MLP}_i\big(\mathrm{LN}(H'_i)\big), \tag{2}$$

where H'_i denotes the intermediate hidden states after the MHA block.

Block-wise Pruning. Let $\mathcal{B} = \{b_1, \ldots, b_{2L}\}$ denote the set of all pruning units. Block-wise pruning aims to remove a subset of units while preserving downstream performance. Existing methods commonly estimate block importance using criteria such as BI, which measures local representational change through the average cosine distance between a block's input and output hidden states. For a block b_i, its BI score [29] can be written as $\mathrm{BI}_i = 1 - \mathbb{E}_{x,t}\left[\frac{(H^{\mathrm{in}}_{b_i,t})^\top H^{\mathrm{out}}_{b_i,t}}{\|H^{\mathrm{in}}_{b_i,t}\|_2 \, \|H^{\mathrm{out}}_{b_i,t}\|_2}\right]$, where the expectation is taken over samples x and token positions t. Another widely used criterion is PPL [32], which evaluates teacher-forced next-token predictive quality after masking or removing candidate blocks, where $\mathrm{PPL} = \exp\big(-\frac{1}{n}\sum_{t=1}^{n} \log p_\theta(w_t \mid w_{<t})\big)$.

2.2 Empirical Study

To better understand how generation drift relates to module redundancy, we conduct an empirical study on GSM8K [38] and HellaSwag [39], which serve as representative benchmarks for generative reasoning and classification, using Llama-3.1-8B-Instruct [40] and Qwen2.5-7B-Instruct [41] as dense models. We quantify generation drift using the forward Kullback–Leibler (KL) divergence from the dense model's next-token distribution to that of the pruned model, where both models are conditioned on the same prefix generated by the dense model. Fig. 2(left) shows that generation drift becomes more pronounced at later decoding positions, suggesting that later prefixes can expose larger predictive discrepancies. This motivates using tail-position drift as a proxy for assessing pruning damage.

Finding 1. Generation drift varies substantially across pruning units, motivating block-level pruning. Fig. 2(middle and right) shows that pruning different layers, MHA blocks, or MLP blocks leads to very different amounts of generation drift. In both models, early layers and the final layer generally induce much larger drift than most middle layers, suggesting that they are more important for stable autoregressive generation, while units causing only limited drift are more likely to be redundant. This broad layer-wise pattern is consistent with prior observations of structural redundancy [29]; our analysis characterizes it through generation drift at finer MHA/MLP granularity. Notably, this difference can also appear within the same layer: some MLP blocks induce much larger drift than their corresponding MHA blocks. This highlights the limitation of coarse-grained layer pruning, which may overlook truly redundant blocks. Appendix B further analyzes the block-level pruning patterns selected by GDPruner.

Finding 2. Low-drift modules are strong candidates for broad redundancy. For each block, we remove it from the dense model and measure both the induced generation drift and the downstream performance drop. Specifically, Fig. 3 reports 64 independent MHA/MLP sublayer ablations on Llama-3.1-8B-Instruct across layer indices 0–31, with drift averaged over positions 65–128 of a 128-token generated continuation. Single-block removal causes only limited distinguishable changes on classification tasks, suggesting that short-horizon decision accuracy may not sufficiently expose functional differences among modules. In contrast, generative reasoning is more sensitive to block removal: low tail drift usually corresponds to minor degradation, whereas high tail drift often leads

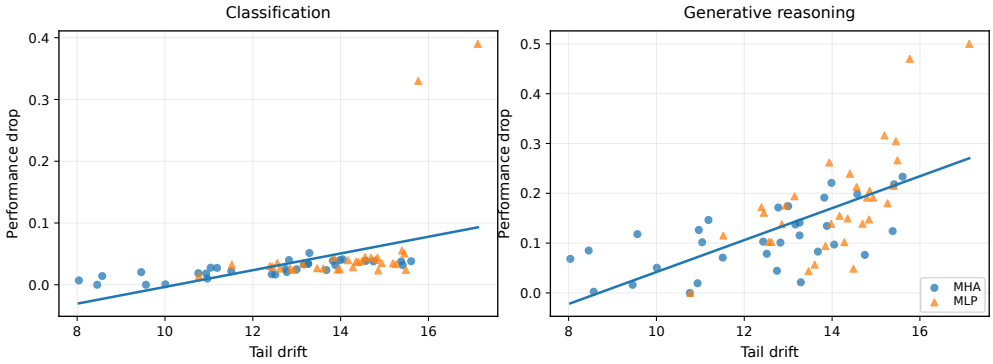


Figure 3: Correlation between generation drift and performance drop.

to substantial performance drops. This indicates that tail-position generation drift better captures whether a block is critical for maintaining stable autoregressive dynamics. These results suggest that generation drift provides a more discriminative signal of module importance, and that low-drift blocks are strong candidates for broad redundancy under the probed generation distribution.

To summarize, the above analysis motivates three design questions: how to construct lightweight and representative generation trajectories, how to exploit generation drift efficiently as a pruning criterion, and how to model dependencies among pruning units during block pruning.

3 Proposed Method

To address these questions, we propose GDPruner, a three-stage block-pruning framework for LLMs, as illustrated in Fig. 4. GDPruner first constructs lightweight self-generated probe trajectories without external calibration data, then evaluates candidate pruning subsets using tail-position generation drift, and finally performs interaction-aware adaptive search to identify broadly redundant blocks.

3.1 Generation Drift

We define generation drift as the divergence between the next-token distributions of the dense and pruned models conditioned on the same autoregressive prefix. Given an input prompt x and a prefix $y_{<t} = (y_1, \ldots, y_{t-1})$, let $p_t(\cdot) = p_\theta(\cdot \mid x, y_{<t})$ denote the next-token distribution of the dense model at decoding step t. For a pruned model obtained by removing a subset of units $\mathcal{S} \subseteq \mathcal{B}$, let $q_t^{(\mathcal{S})}(\cdot) = q_\theta^{(\mathcal{S})}(\cdot \mid x, y_{<t})$ denote the next-token distribution of the corresponding pruned model under the same prefix. We measure the step-wise generation drift by the dense-to-pruned KL divergence:

$$\Delta_t^{(\mathcal{S})} = D_{\mathrm{KL}}\left(p_t \,\|\, q_t^{(\mathcal{S})}\right) = \sum_{v \in \mathcal{V}} p_t(v) \log \frac{p_t(v)}{q_t^{(\mathcal{S})}(v)}, \tag{3}$$

where $\mathcal{V}$ is the vocabulary. A smaller $\Delta_t^{(\mathcal{S})}$ indicates that pruning introduces less predictive mismatch at step t, serving as a proxy for lower pruning risk during autoregressive generation.

This criterion is aligned with trajectory-level preservation. Let T denote the trajectory length. Let $P_\theta(y_{1:T} \mid x)$ denote the autoregressive distribution over complete trajectories generated by the dense model, and let $Q_\theta^{(\mathcal{S})}(y_{1:T} \mid x)$ denote the corresponding trajectory distribution induced by the pruned model. By the chain rule of KL divergence,

$$D_{\mathrm{KL}}\left(P_\theta(\cdot \mid x) \,\|\, Q_\theta^{(\mathcal{S})}(\cdot \mid x)\right) = \sum_{t=1}^{T} \mathbb{E}_{y_{<t} \sim P_\theta(\cdot|x)} \left[D_{\mathrm{KL}}\left(p_\theta(\cdot \mid x, y_{<t}) \,\|\, q_\theta^{(\mathcal{S})}(\cdot \mid x, y_{<t})\right)\right]. \tag{4}$$

Thus, the step-wise KL can be viewed as a local contribution to the full dense–pruned trajectory divergence. In practice, we estimate it on dense-generated shared prefixes sampled from P_θ, which avoids stochastic branching and enables fair comparison under identical trajectories. The dense-to-pruned direction penalizes low probability on dense-preferred tokens, whose deviations may accumulate during autoregressive generation.

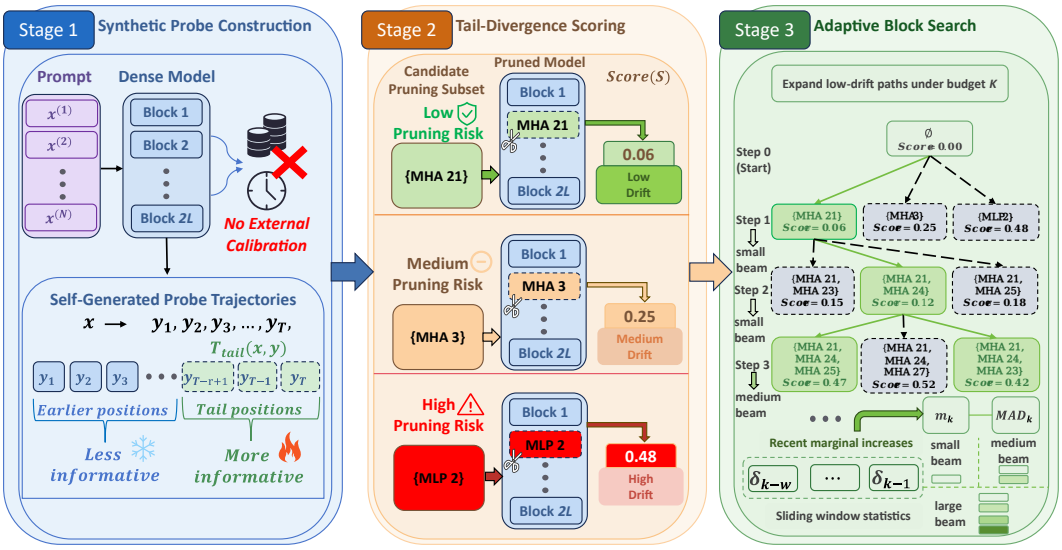


Figure 4: Overview of GDPruner. It consists of three stages: synthetic probe construction, tail-divergence scoring, and adaptive block search to identify prunable modules.

3.2 Synthetic Probe Construction

In the first stage, GDPruner constructs the probe set used for drift estimation. We use a small set of generic seed prompts that are independent of downstream evaluation tasks and external calibration corpora. Given the prompt set $\mathcal{X} = \{x^{(n)}\}_{n=1}^{N}$, the dense model autoregressively generates a continuation $y^{(n)} = (y_1^{(n)}, \ldots, y_T^{(n)})$ for each prompt $x^{(n)}$. This forms a probe set $\mathcal{P} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, where all evaluated tokens are generated by the dense model itself. Therefore, GDPruner estimates pruning-induced generation drift without relying on external calibration data.

Motivated by Sec. 2.2, GDPruner focuses on tail decoding positions, where pruning-induced discrepancies are more pronounced. For each probe trajectory (x, y) with length T, we define the tail-position index set as

$$\mathcal{T}_{\text{tail}}(x, y) = \{T - r + 1, \ldots, T\}, \tag{5}$$

where r is the tail length. These trajectories and their corresponding tail positions are reused throughout the pruning search.

3.3 Tail-Divergence Scoring

Given a candidate pruning subset $\mathcal{S} \subseteq \mathcal{B}$, GDPruner ranks candidate subsets by averaging the step-wise drift defined above over the tail positions of the synthetic probe trajectories. Since later positions expose larger pruning-induced discrepancies, we evaluate candidate subsets only on the tail decoding region of each probe trajectory. For a probe trajectory $(x, y) \in \mathcal{P}$, let $\mathcal{T}_{\text{tail}}(x, y)$ denote its tail-position index set defined in Sec. 3.2. The trajectory-level drift of $\mathcal{S}$ is computed as

$$\Delta_{\text{tail}}^{(\mathcal{S})}(x, y) = \frac{1}{|\mathcal{T}_{\text{tail}}(x, y)|} \sum_{t \in \mathcal{T}_{\text{tail}}(x,y)} \Delta_t^{(\mathcal{S})}. \tag{6}$$

GDPruner then aggregates this quantity over the entire probe set and defines the score of $\mathcal{S}$ as

$$\text{Score}(\mathcal{S}) = \frac{1}{|\mathcal{P}|} \sum_{(x,y) \in \mathcal{P}} \Delta_{\text{tail}}^{(\mathcal{S})}(x, y). \tag{7}$$

In the second stage, GDPruner interprets the tail-drift score as the pruning risk of a candidate subset. Lower scores indicate smaller deviation from the dense model and therefore higher suitability for pruning. GDPruner prioritizes low-score subsets and avoids removing blocks that induce large tail-position drift. In practice, we cache the dense model's tail distributions once and reuse them across candidate evaluations to reduce search cost.

3.4 Adaptive Block Search

In the final stage, GDPruner searches for a pruning subset under a target sparsity budget of K blocks. Starting from the empty subset, it expands candidates by adding one block at each step. Since block effects are interdependent, independent ranking can be suboptimal. GDPruner therefore performs subset-level search: candidates are scored by the tail-drift objective in Sec. 3.3, and low-drift subsets are retained for further expansion. This prioritizes low-risk pruning paths while discarding candidates that already induce large generation drift.

To balance search quality and efficiency, GDPruner uses an adaptive beam width. The beam starts small and expands only when pruning becomes more sensitive. Let s_k denote the best retained score at step k, i.e., the lowest tail-drift score among the retained candidates. We define the current marginal score change as $\delta_k = s_k - s_{k-1}$, and compare it with recent marginal increases stored in a sliding window. The anomaly score is computed as

$$z_k = \frac{\delta_k - m_k}{\mathrm{MAD}_k + \epsilon}, \tag{8}$$

where m_k and MAD_k are the median and median absolute deviation (MAD) of recent marginal increases within the sliding window, and ϵ is a small constant for numerical stability.

The beam width for the next step is then adjusted according to the severity of the current increase:

$$B_{k+1} = \begin{cases} B_{\min}, & z_k < \tau_1, \\ B_{\text{mid}}, & \tau_1 \le z_k < \tau_2, \\ B_{\max}, & z_k \ge \tau_2, \end{cases} \tag{9}$$

where $B_{\min} < B_{\text{mid}} < B_{\max}$ denote the small, medium, and large beam widths, and τ_1, τ_2 are the corresponding expansion thresholds. When the marginal drift change is mild, GDPruner keeps a small beam for efficiency; when the drift rises sharply, it expands the beam to explore more candidate subsets and reduce the risk of pruning generation-critical blocks.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks. We evaluate GDPruner primarily on two representative decoder-only LLMs: Llama-3.1-8B-Instruct [40] and Qwen2.5-7B-Instruct [41]. Unless otherwise specified, all experiments are conducted on a single NVIDIA RTX A5000 GPU with 24GB memory. To evaluate GDPruner from both conventional accuracy preservation and generation robustness perspectives, we use a diverse set of classification and generative reasoning benchmarks. The classification group includes commonsense and knowledge-intensive multiple-choice tasks: Winogrande [42], PIQA [43], HellaSwag [39], ARC-Easy and ARC-Challenge [44], and MMLU [45]. The generative reasoning group includes GSM8K [38] for multi-step mathematical reasoning, Math500 [46] for mathematical generation, and HumanEval [47] for code generation. Appendix C provides details of the datasets.

Baselines. We compare GDPruner with representative structured pruning baselines, including **ShortGPT** [29], **SLEB** [31], **LLMDrop** [48], and **BlockPruner** [32]. ShortGPT prunes layers based on BI, while SLEB is a layer-level method that combines structured pruning with calibration-based evaluation. LLMDrop extends pruning to the block level using local input–output similarity, and BlockPruner measures block importance by masking candidate blocks and evaluating the resulting PPL change on a calibration set. These baselines cover layer and block pruning, as well as local-discrepancy-based and calibration-based scoring strategies, enabling a comprehensive comparison. For all baselines, we follow the original implementations or descriptions as closely as possible and adapt them to our setting when necessary. More implementation details are provided in Appendix E.

4.2 Overall Performance

Prior studies [32] commonly restrict the pruning ratio to approximately 25%. Consistent with these settings, we also limit the pruning ratio to this range in our experiments. Table 1 reports results across multiple pruning ratios on two models. Overall, GDPruner achieves strong performance

Table 1: Performance comparison of structured pruning methods on **Llama-3.1-8B-Instruct** and **Qwen2.5-7B-Instruct** under different pruning ratios. AVG denotes the unweighted average over all tasks, and RP denotes the relative performance with respect to the dense model. The best and second-best results within each pruning-ratio group are marked in **bold** and underlined, respectively.

Model	Method	Pruning Unit	Pruning Ratio	Wino.	PIQA	HellaSwag	ARC-e	ARC-c	MMLU	GSM8K	Math500	HumanEval	AVG	RP
Llama-3.1-8B-Instruct	Dense	–	0.00%	73.48	81.39	79.51	79.76	55.80	68.38	78.39	36.40	59.15	68.03	100.0
	ShortGPT	Layer	10.00%	74.19	78.67	76.76	77.57	52.39	67.34	67.63	23.20	28.05	60.64	89.1
	SLEB	Layer	10.00%	68.82	78.29	73.43	71.63	46.16	48.98	18.12	13.40	31.71	50.06	73.6
	LLMDrop	Block	10.00%	75.53	80.25	78.54	78.96	54.01	67.62	71.42	**27.60**	**48.78**	**64.75**	**95.2**
	BlockPruner	Block	10.00%	**75.69**	**80.79**	**78.68**	**79.50**	**55.38**	67.86	72.93	26.40	35.37	63.62	93.5
	Ours	Block	10.00%	74.43	79.71	77.77	78.91	54.52	**67.97**	**74.68**	27.00	40.24	63.91	93.9
	ShortGPT	Layer	15.00%	70.56	74.59	71.23	68.69	49.49	67.31	20.39	5.00	10.37	48.63	71.5
	SLEB	Layer	15.00%	68.67	77.15	70.37	66.79	41.89	48.99	12.96	7.80	18.29	45.88	67.4
	LLMDrop	Block	15.00%	**75.53**	**79.27**	**77.32**	76.47	**53.67**	**67.46**	**63.46**	17.40	**23.17**	**59.31**	**87.2**
	BlockPruner	Block	15.00%	73.09	79.16	76.42	74.62	49.74	62.82	49.20	14.80	15.85	55.08	81.0
	Ours	Block	15.00%	72.14	77.97	75.16	**76.94**	50.68	62.68	53.22	**21.60**	**23.17**	57.06	83.9
	ShortGPT	Layer	20.00%	71.27	74.27	69.30	63.13	44.71	**64.93**	9.86	1.60	4.27	44.82	65.9
	SLEB	Layer	20.00%	66.85	75.90	68.38	64.44	41.13	47.54	11.37	6.20	12.80	43.85	64.5
	LLMDrop	Block	20.00%	**73.95**	76.71	**75.22**	70.24	48.29	38.23	13.27	4.80	1.83	44.73	65.8
	BlockPruner	Block	20.00%	69.61	76.99	71.84	73.06	47.61	55.77	**33.28**	9.60	6.71	49.39	72.6
	Ours	Block	20.00%	69.46	**77.42**	73.14	**75.00**	**49.15**	56.47	30.48	**11.40**	**15.85**	**50.93**	**74.9**
	ShortGPT	Layer	25.00%	69.14	70.84	61.90	53.37	40.44	**65.78**	0.53	0.40	0.00	40.27	59.2
	SLEB	Layer	25.00%	59.75	73.29	60.12	57.95	34.81	33.72	1.59	3.20	6.10	36.73	54.0
	LLMDrop	Block	25.00%	**71.27**	74.27	**70.90**	63.34	42.75	34.59	5.53	3.20	0.61	40.72	59.9
	BlockPruner	Block	25.00%	65.67	**75.68**	67.78	65.99	41.89	42.08	3.11	4.60	1.22	40.89	60.1
	Ours	Block	25.00%	68.82	74.92	68.40	**72.10**	**46.25**	49.91	**12.51**	**7.20**	**7.93**	**45.34**	**66.6**
Qwen2.5-7B-Instruct	Dense	–	0.00%	70.64	80.36	80.48	81.06	54.86	71.78	83.62	44.40	64.63	70.20	100.0
	ShortGPT	Layer	10.00%	62.59	**80.03**	73.62	**75.38**	**52.73**	52.06	39.88	23.60	26.22	54.01	76.9
	SLEB	Layer	10.00%	63.77	77.75	70.58	74.71	49.74	54.37	29.57	20.20	29.27	52.22	74.4
	LLMDrop	Block	10.00%	64.40	79.60	**77.67**	69.57	46.93	**69.44**	15.47	11.00	26.22	51.14	72.9
	BlockPruner	Block	10.00%	**67.01**	79.11	76.19	74.28	50.51	62.56	42.15	29.40	36.59	57.53	82.0
	Ours	Block	10.00%	65.82	78.62	77.61	75.29	51.79	67.39	**49.96**	**29.60**	**46.95**	**60.34**	**85.9**
	ShortGPT	Layer	15.00%	59.19	**79.11**	71.01	72.77	49.40	45.46	15.47	14.60	18.29	47.26	67.3
	SLEB	Layer	15.00%	60.46	77.42	69.79	71.46	47.87	48.11	14.25	14.60	21.95	47.32	67.4
	LLMDrop	Block	15.00%	62.04	76.66	71.31	59.72	40.53	59.47	0.00	1.20	0.61	41.28	58.8
	BlockPruner	Block	15.00%	61.01	78.84	72.56	71.76	46.25	48.63	6.22	13.00	21.95	46.69	66.5
	Ours	Block	15.00%	**63.06**	78.40	**76.16**	**75.04**	**51.62**	**62.75**	**24.41**	**25.00**	**35.37**	**54.65**	**77.8**
	ShortGPT	Layer	20.00%	56.91	**77.04**	64.83	63.85	38.05	29.83	3.34	5.00	9.15	38.67	55.1
	SLEB	Layer	20.00%	57.62	75.08	63.92	65.11	40.44	28.96	3.41	6.20	11.59	39.15	55.8
	LLMDrop	Block	20.00%	55.64	73.94	65.82	53.41	37.97	28.79	0.00	0.60	0.00	35.13	50.0
	BlockPruner	Block	20.00%	57.06	76.99	66.81	58.08	37.54	42.95	5.16	7.60	17.07	41.03	58.4
	Ours	Block	20.00%	**60.38**	76.88	**71.81**	**71.84**	**51.28**	**53.62**	**13.57**	**18.00**	**26.83**	**49.36**	**70.3**
	ShortGPT	Layer	25.00%	51.22	73.94	60.16	58.75	37.88	26.28	2.27	3.60	6.71	35.65	50.8
	SLEB	Layer	25.00%	54.62	73.39	59.55	59.81	37.29	26.56	2.43	3.20	9.76	36.29	51.7
	LLMDrop	Block	25.00%	51.22	67.57	50.08	41.37	33.62	23.69	0.00	0.60	0.00	29.79	42.4
	BlockPruner	Block	25.00%	54.30	75.90	63.31	67.89	39.25	29.51	**3.34**	4.80	11.59	38.88	55.4
	Ours	Block	25.00%	**56.51**	**76.61**	**66.62**	**69.36**	**47.18**	**38.32**	3.11	**7.80**	**15.85**	**42.37**	**60.4**

retention, and its advantage becomes more pronounced at higher pruning ratios. Although some block-level baselines remain competitive under mild pruning, their performance becomes less stable as the pruning ratio increases, especially on generation-intensive tasks. This indicates that generation-drift-based scoring is particularly effective when pruning becomes more aggressive and the risk of removing generation-critical blocks increases. While several baselines preserve classification benchmarks reasonably well, their degradation is more evident on generative reasoning tasks such as GSM8K, Math500, and HumanEval. In contrast, GDPruner better preserves these tasks by favoring blocks whose removal does not substantially disturb the dense model's autoregressive trajectory, leading to a stronger balance between classification performance and generative reasoning ability. Appendix F reports additional results on five further LLMs spanning different families, scales, and architectures, including an MoE model. Matched LoRA recovery results are provided in Appendix G, and qualitative generation examples are shown in Appendix I.

4.3 Efficiency Analysis

We provide an integrated efficiency–performance analysis on two models at a 20% pruning ratio. Table 2 shows that GDPruner achieves a strong efficiency–performance trade-off. While some baselines obtain comparable latency or throughput, they often suffer larger performance drops, particularly on generative reasoning benchmarks. GDPruner instead maintains competitive acceleration while preserving the best overall and generative accuracy, indicating that its generation-drift criterion removes blocks that are not only computationally redundant but also less critical to autoregressive generation. These results demonstrate that GDPruner provides a practical balance between inference efficiency and generation robustness. The offline pruning cost is reported in Appendix H.

Table 2: Efficiency–performance comparison at 20% pruning ratio on two LLMs. Gen. Acc. denotes the average accuracy on GSM8K, Math500, and HumanEval.

Model	Method	AVG ↑	Gen. Acc. ↑	Latency ↓	Tok/s ↑	Speedup ↑
Llama-3.1-8B Instruct	Dense	68.03	57.98	3.9955	32.04	1.000×
	ShortGPT	44.82	5.24	3.5090	36.48	1.139×
	SLEB	43.85	10.12	**3.2713**	**39.13**	**1.221**×
	LLMDrop	44.73	6.63	3.3956	37.70	1.177×
	BlockPruner	49.39	16.53	3.3738	37.94	1.184×
	Ours	**50.93**	**19.24**	3.3057	38.72	1.209×
Qwen2.5-7B Instruct	Dense	70.20	64.22	3.6379	35.19	1.000×
	ShortGPT	38.67	5.83	3.0605	41.82	1.189×
	SLEB	39.15	7.07	3.1475	40.67	1.156×
	LLMDrop	35.13	0.20	3.1779	40.28	1.145×
	BlockPruner	41.03	9.94	3.0269	42.29	1.202×
	Ours	**49.36**	**19.47**	**3.0012**	**42.65**	**1.212**×

Table 3: Ablation study results on Llama-3.1-8B-Instruct at 20% pruning ratio.

Method	AVG
GDPruner	**50.93**
w/o self-generated probes	42.79 (−**8.14**)
w/o tail-position drift scoring	46.56 (−**4.37**)
w/o KL divergence	49.67 (−**1.26**)
w/o adaptive block search	48.28 (−**2.65**)

Table 4: Ablation of discrepancy metrics at 20% pruning ratio. Gen. Acc. denotes the average accuracy on GSM8K, Math500, and HumanEval.

Model	Metric	Wino.	PIQA	Hella.	ARC-e	ARC-c	MMLU	GSM8K	Math500	HumanEval	AVG	Gen. Acc.
Llama-3.1 8B-Instruct	NLL	68.43	**77.97**	72.03	73.44	47.78	49.64	16.76	8.40	14.63	47.68	13.26
	Entropy	68.19	77.31	71.04	70.54	46.42	46.78	18.35	9.80	**17.07**	47.28	15.07
	JS	69.06	77.31	**74.02**	74.58	**49.91**	55.86	21.08	**14.20**	10.98	49.67	15.42
	KL	**69.46**	77.42	73.14	**75.00**	49.15	**56.47**	**30.48**	11.40	15.85	**50.93**	**19.24**
Qwen2.5 7B-Instruct	NLL	58.01	74.54	66.29	63.76	40.70	44.66	3.87	5.00	13.41	41.14	7.43
	Entropy	56.20	72.42	68.29	68.14	45.14	49.05	6.60	9.80	18.29	43.77	11.56
	JS	59.27	76.82	70.13	70.54	47.27	53.26	8.11	11.40	24.39	46.80	14.63
	KL	**60.38**	**76.88**	**71.81**	**71.84**	**51.28**	**53.62**	**13.57**	**18.00**	**26.83**	**49.36**	**19.47**

4.4 Ablation Study

Table 3 evaluates the contribution of key design choices in GDPruner. Overall, removing any component leads to consistent degradation, confirming that the final performance is not driven by a single heuristic but by the combination of generation-aligned probing, tail scoring, and adaptive search. The largest drop comes from replacing self-generated probes with random probes, showing that the quality of probe trajectories is critical for exposing generation drift. Since self-generated trajectories follow the dense model's own decoding behavior, they better reveal which blocks are important for maintaining stable autoregressive dynamics. Tail-only scoring is also important: averaging over the full sequence weakens the pruning signal because early positions are less affected by accumulated errors, while tail positions provide a more discriminative view of long-horizon generation stability. We further examine the drift discrepancy metric and search strategy in detail.

Effects of discrepancy metrics for drift estimation. Table 4 further compares different discrepancy metrics for drift-based pruning. KL divergence achieves the best overall performance and generative accuracy on both models, with Jensen–Shannon (JS) divergence serving as the second-best alternative on these aggregate metrics. This suggests that explicitly comparing the dense and pruned output distributions is important. In our setting, KL divergence is computed from the dense model to the pruned model. It strongly penalizes cases where the pruned model assigns low probability to tokens that the dense model considers likely. This property is well aligned with structured pruning, where the goal is not merely to produce confident predictions, but to preserve the dense model's predictive distributions on self-generated prefixes after removing blocks.

By contrast, JS divergence symmetrizes and smooths the mismatch between the two distributions, making it less sensitive to one-sided failures where the pruned model loses probability mass on dense-preferred tokens. Negative log-likelihood (NLL) criteria only evaluate the realized token and ignore the broader next-token distribution, potentially making them less informative for identifying modules that affect alternative reasoning paths. Entropy-based criteria measure the uncertainty of the pruned model itself, but do not indicate whether its uncertainty or confidence is aligned with the dense model. Together, these results support dense-to-pruned KL as an effective pruning signal for retaining overall and generative performance in the evaluated settings. Appendix D further discusses these metrics and why KL divergence is well aligned with generation-drift estimation.

Table 5: Sensitivity to probe length T and tail length r on Llama-3.1-8B-Instruct at 20% pruning. MC Avg. and Gen. Avg. denote the average accuracy over the six multiple-choice tasks and the three generative tasks, respectively; bold marks the best setting within each probe length.

Probe length T	Tail length r	MC Avg.	Gen. Avg.	Overall Avg.
128	16	65.63	18.49	49.92
	32	65.12	15.20	48.48
	64	**66.77**	**19.24**	**50.93**
	128	64.57	10.55	46.56
512	64	63.92	10.83	46.22
	128	**66.76**	**17.21**	**50.24**
1024	128	**66.54**	**15.04**	**49.37**
	256	66.50	12.98	48.66

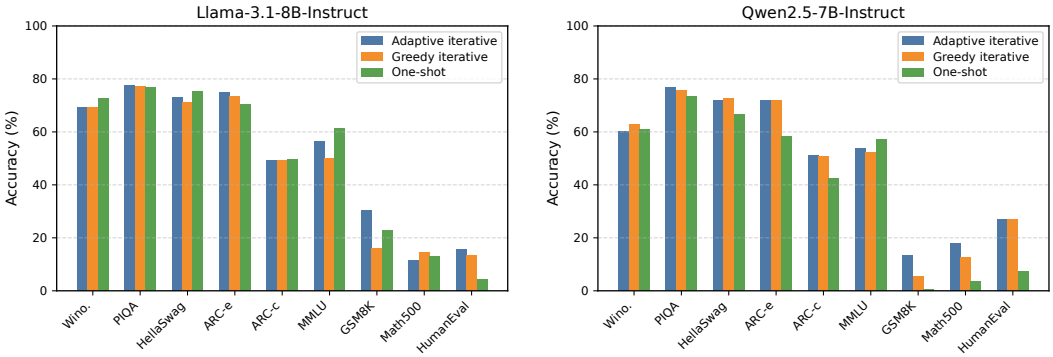


Figure 5: Impact of block pruning strategy at 20% pruning ratio.

Effects of block search strategy. As shown in Fig. 5, one-shot ranking and greedy iterative search remain competitive at 20% pruning, indicating that generation drift already provides an effective redundancy signal. One-shot ranking evaluates blocks independently and ignores interactions among removed blocks, while greedy iterative search updates decisions step by step but remains locally myopic. Adaptive search better accounts for such block dependencies, which is especially important for generative reasoning tasks where pruning-induced errors can accumulate across decoding steps. These results suggest that generation drift provides the core pruning signal, while adaptive search helps improve pruning quality by avoiding harmful block combinations.

Sensitivity to probe and tail lengths. Table 5 shows that longer probes and wider tail windows do not consistently improve retention. Extending a probe changes the generated states being evaluated, whereas enlarging the tail changes which positions contribute to the pruning score. The best-performing tail windows remain compact as the generation horizon grows, while generative accuracy is more sensitive to tail selection than multiple-choice accuracy. These observations support focusing on informative late-position discrepancies rather than scaling the scoring window proportionally with the trajectory. Additional analyses of probe count and adaptive-beam thresholds are provided in Appendix F. Overall, these results confirm that compact self-generated probes and targeted tail scoring are sufficient for effective generation-aware pruning.

5 Related Work

Recent studies suggest that LLMs exhibit substantial structural redundancy, particularly in deeper layers [28–30, 49]. This observation has motivated structured pruning methods that remove entire computational units rather than individual weights alone. Among these approaches, layer pruning is a simple and effective strategy for improving efficiency by directly reducing model depth. ShortGPT estimates layer redundancy using input–output similarity, whereas LaCo compresses the network by merging adjacent layers to reduce its effective depth [29, 30]. ShortGPT also observes that

pruned models can suffer much larger degradation on generative reasoning tasks and accordingly proposes ShortGPT-gen, in which input tokens skip redundant layers while generated tokens still pass through all layers. Although this design alleviates error accumulation during generation, it also makes it difficult to realize the full acceleration and compression benefits of pruning in practical deployment. SLEB further combines structured compression with calibration-based evaluation within a more efficient pruning pipeline [31]. Although such methods can yield clear acceleration gains, layer pruning is inherently coarse-grained and may remove useful subcomponents along with redundant ones. This limitation has led to growing interest in block-level pruning, in which MHA and MLP sublayers are selectively removed to achieve finer control over the speed–accuracy trade-off [32, 34, 48].

Existing block-pruning methods differ primarily in how they estimate block importance. LLMDrop extends BI-based pruning to the block level by using local input–output similarity as a redundancy signal [48]. FinerCut evaluates changes in model outputs after removing individual MHA or MLP sublayers, considering norm-, arccos-, and JS-based criteria for block ranking [34]. BlockPruner evaluates candidate blocks by masking them and measuring the resulting change in perplexity on a calibration set [32]. FLAP uses calibration-based structural sensitivity to guide pruning decisions [25]. EntroDrop instead employs entropy-based statistics as the pruning signal to capture compression-sensitive hidden-state dynamics more efficiently [50]. IG-Pruning dynamically selects input-conditioned block masks at inference time, providing an input-adaptive alternative to static block pruning [51]. Although these methods differ in formulation, they primarily rely on local representations, output discrepancies, calibration-based criteria, or input-adaptive routing. In contrast, GDPruner identifies a static pruning mask using tail-position KL divergence on shared dense-generated prefixes as a proxy for autoregressive generation instability.

6 Conclusion and Limitations

In this paper, we study generation drift in structured pruning for LLMs. Our analysis shows that discrepancies become more pronounced at later decoding positions and vary substantially across layers and blocks, suggesting that module redundancy cannot be fully captured by short-horizon classification behavior or coarse layer-level criteria. Motivated by this finding, we propose GDPruner, a calibration-data-free block pruning framework that uses self-generated trajectories and generation drift to identify strong candidates for broad redundancy. Experiments across multiple LLMs and benchmarks show that GDPruner achieves stronger overall performance than existing structured pruning baselines in most settings while better preserving complex generative reasoning ability. By reducing inference cost with better relative retention of generative performance, GDPruner may improve the accessibility and efficiency of LLM deployment.

GDPruner avoids external calibration corpora by estimating generation drift from self-generated probe trajectories, improving efficiency and reducing sensitivity to calibration-data selection. However, this makes the pruning signal depend on the dense model's own generation quality. If the dense model produces inaccurate or insufficiently diverse trajectories, GDPruner may preserve its behavior without fully capturing task-correct or domain-relevant generation.

Acknowledgements and disclosures

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University); No. RS-2024-00457882, AI Research Hub Project; No. RS-2025-02304828, Artificial Intelligence Star Fellowship Support Program to Nurture the Best Talents; and No. RS-2026-25507543, Development of AI Co-Scientist based on Scientific Causal World Model).

Competing interests. The authors declare no competing interests.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 6000–6010. Curran Associates, Inc., 2017.

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

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with Pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc., 2022.

[6] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

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

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[9] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10323–10337, 2023.

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

[11] Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. APT: Adaptive pruning and tuning pretrained language models for efficient training and inference. In *Proceedings of the 41st International Conference on Machine Learning*, pages 60812–60831, 2024.

[12] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 38087–38099, 2023.

[13] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100, 2024.

[14] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc., 2022.

[15] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36, pages 4396–4429. Curran Associates, Inc., 2023.

[16] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024.

[17] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

[18] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174. Association for Computational Linguistics, 2020.

[19] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 24872–24895, 2024.

[20] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc., 2023.

[21] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey. *Transactions on Machine Learning Research*, 2024.

[22] Eldar Kurtic, Elias Frantar, and Dan Alistarh. ZipLM: Inference-aware structured pruning of language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 65597–65617. Curran Associates, Inc., 2023.

[23] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, 2024.

[24] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024.

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

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

[27] Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 7736–7758. Curran Associates, Inc., 2024.

[28] Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. Search for efficient large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 139294–139315. Curran Associates, Inc., 2024.

[29] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204. Association for Computational Linguistics, 2025.

[30] Yifei Yang, Zouying Cao, and Hai Zhao. LaCo: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417. Association for Computational Linguistics, 2024.

[31] Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: Streamlining LLMs through redundancy verification and elimination of transformer blocks. In *Proceedings of the 41st International Conference on Machine Learning*, pages 46136–46155, 2024.

[32] Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. BlockPruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5065–5080. Association for Computational Linguistics, 2025.

[33] Haihang Wu. LLM-BIP: Structured pruning for large language models with block-wise forward importance propagation. *arXiv preprint arXiv:2412.06419*, 2024.

[34] Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. FinerCut: Finer-grained interpretable layer pruning for large language models. In *Workshop on Machine Learning and Compression, NeurIPS 2024*, 2024.

[35] Tae-Won Kim, Hee-Dong Kim, Jumin Han, and Seong-Whan Lee. Conditioning activation interventions on evolving reasoning states in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 46826–46846. Association for Computational Linguistics, 2026.

[36] Ava Spataru, Eric Hambro, Elena Voita, and Nicola Cancedda. Know when to stop: A study of semantic drift in text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3656–3671. Association for Computational Linguistics, 2024.

[37] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pages 10524–10533, 2020.

[38] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[39] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. Association for Computational Linguistics, 2019.

[40] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[41] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[42] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[43] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

[44] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[45] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

[46] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Curran Associates, Inc., 2021.

[47] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[48] Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. Uncovering the redundancy in transformers via a unified study of layer dropping. *Transactions on Machine Learning Research*, 2026.

[49] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. In *International Conference on Learning Representations*, 2025.

[50] Liangwei Yang, Yuhui Xu, Juntao Tan, Doyen Sahoo, Silvio Savarese, Caiming Xiong, Huan Wang, and Shelby Heinecke. Entropy-based block pruning for efficient large language models. In *International Conference on Learning Representations*, 2026.

[51] Kangyu Qiao, Shaolei Zhang, and Yang Feng. IG-Pruning: Input-guided block pruning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10618–10629. Association for Computational Linguistics, 2025.

[52] Woo-Jeoung Nam, Shir Gur, Jaesik Choi, Lior Wolf, and Seong-Whan Lee. Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2501–2508, 2020.

[53] Hee-Dong Kim, Yeong-Joon Ju, Jung-Ho Hong, and Seong-Whan Lee. CIRF: Importance of related features for plausible counterfactual explanations. *Information Sciences*, 678:120974, 2024.

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

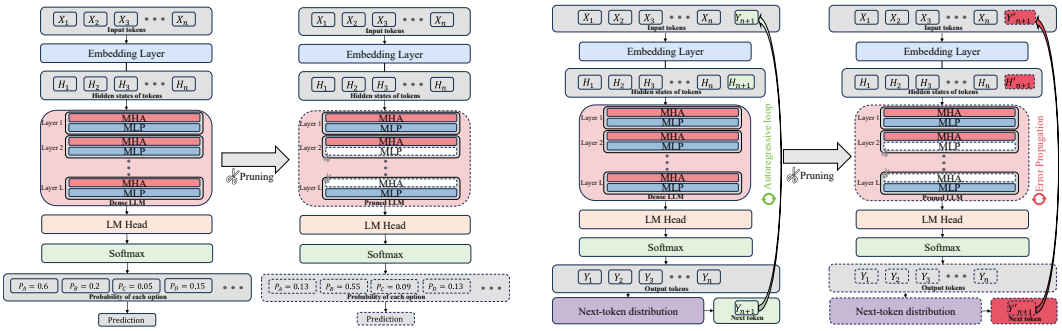
[55] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[56] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

[57] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[58] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc., 2023.

[59] Jumin Han, Jun-Hee Kim, and Seong-Whan Lee. Towards generalizable 3D human pose estimation via ensembles on flat loss landscapes. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.



(a) Classification prediction (b) Autoregressive generative reasoning

Figure 6: Illustration of how pruning affects classification prediction and autoregressive generative reasoning. In classification-style prediction, pruning changes the final option probabilities within a single forward pass. In autoregressive generation, pruning-induced deviations can be fed back as future inputs, causing errors to propagate across decoding steps.

A Illustration of Error Accumulation under Autoregressive Pruning

Fig. 6 illustrates why structured pruning can affect classification-style prediction and autoregressive generation in different ways. For classification prediction, the model takes the input tokens, computes hidden states through the pruned network, and produces option probabilities in a single forward pass. Although pruning may perturb hidden representations and change the probability assigned to each option, the prediction process ends after this step. Therefore, the pruning-induced perturbation is not recursively fed back into the model, and its effect is mainly confined to the current computation path.

In contrast, autoregressive generation forms a closed-loop process. The model first predicts the next-token distribution, samples or selects the next token, and then feeds this token back as part of the input for the following decoding step. After pruning, a small deviation in hidden states or next-token probabilities can change the generated token. Once a different token is produced, subsequent hidden states are computed from a different context, which further enlarges the discrepancy between the dense and pruned models. As a result, the error introduced at an early decoding step may be carried forward and amplified in later steps.

This difference motivates our use of tail-position generation drift as a proxy for pruning risk. GDPruner evaluates the dense and pruned models on shared dense-generated prefixes; it does not directly measure divergence between free-running rollouts. Later prefixes can expose larger predictive discrepancies, motivating the use of tail-position distribution mismatch to identify blocks important for stable autoregressive generation.

B Analysis of Pruned Blocks

The pruned blocks selected by GDPruner also provide useful insight into the structural redundancy of different LLMs. In this section, we visualize the retained and removed units at a representative pruning ratio and analyze the resulting patterns across models.

Fig. 7 visualizes the pruned units in Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct at 25% pruning ratio. A first notable observation is that the pruning pattern is highly non-uniform across both depth and block type. Rather than being removed uniformly, pruned units cluster within particular depth ranges, while many early and late units are preserved. This observation is consistent with our earlier finding that the contributions of different pruning units to generation stability are highly heterogeneous.

A second observation is that, under the same pruning budget, GDPruner tends to prune more MHA units than MLP units. This suggests that, at least under moderate structured pruning, attention blocks contain more removable redundancy than MLP blocks, while a larger fraction of MLP blocks remains important for maintaining stable generation. At the same time, the exact pruning pattern differs between models. In Llama-3.1-8B-Instruct, the removed units are more concentrated in several

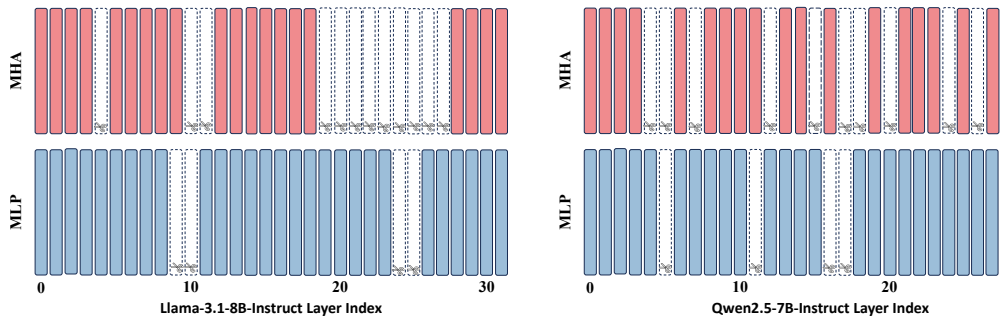


Figure 7: Visualization of pruned blocks at 25% pruning ratio for Llama-3.1-8B-Instruct (left) and Qwen2.5-7B-Instruct (right) using GDPruner.

contiguous mid-to-late depth regions, whereas in Qwen2.5-7B-Instruct the pruning pattern is more distributed across layers. This suggests that although generation-critical redundancy exists in both models, its layer-wise organization remains model-dependent.

Another notable pattern is that GDPruner rarely removes all computation across a broad consecutive depth range. Instead, it preserves a backbone of important units while selectively removing less critical ones. This behavior suggests that effective structured pruning for autoregressive LLMs is not simply a matter of discarding the deepest or most localized blocks, but rather of identifying which specific components can be removed without destabilizing long-horizon generation. In this sense, the observed pruning patterns further support the use of generation drift as a redundancy criterion: the selected subsets reflect both regularities shared across models and architecture-specific differences in how stable generation is maintained.

Overall, these results show that GDPruner not only improves pruning performance but also serves as a useful tool for analyzing the functional organization of LLM blocks, complementing attribution-based analyses of unit contributions [52, 53]. The observed pruning patterns reinforce our earlier empirical findings that redundancy is highly uneven across units, that generation-critical blocks are concentrated in specific regions, and that fine-grained block pruning provides a more flexible and informative alternative to coarse layer pruning.

C Details of Datasets

PIQA [43] is a benchmark for evaluating models' understanding of physical commonsense. It adopts a multiple-choice format in which the model selects the more plausible solution to a goal from two candidate options.

Winogrande [42] is a large-scale commonsense reasoning benchmark designed to reduce annotation artifacts in pronoun resolution tasks. It consists of fill-in-the-blank questions with two candidate answers, requiring the model to select the option that best fits the context of the sentence.

HellaSwag [39] is a commonsense reasoning dataset designed to evaluate models' ability to complete everyday scenarios. It presents a context and several candidate continuations, requiring the model to identify the most plausible ending.

ARC-Easy and **ARC-Challenge** [44] are multiple-choice science question benchmarks derived from grade-school exams. ARC-Easy contains relatively straightforward questions, while ARC-Challenge is more difficult and requires stronger reasoning and knowledge integration.

MMLU [45] is a broad knowledge benchmark that covers a wide range of academic and professional domains. It is designed to evaluate both factual knowledge and reasoning ability through multiple-choice questions.

GSM8K [38] is a benchmark for grade-school mathematical reasoning. It requires the model to generate multi-step solutions to arithmetic word problems and is widely used to evaluate chain-of-thought reasoning ability.

Math500 [46] is a challenging mathematical reasoning benchmark consisting of difficult competition-level math problems. It is commonly used to evaluate models' ability to perform long-form symbolic and quantitative reasoning.

HumanEval [47] is a code generation benchmark that evaluates functional correctness. Given a function signature and a docstring, the model is required to generate code that passes a set of hidden unit tests.

D Discussion of Drift Metrics

In this section, we further discuss the discrepancy metrics used for generation-drift-based pruning. Given a dense next-token distribution p_t and a pruned-model distribution q_t under the same prefix, different metrics emphasize different aspects of the dense–pruned mismatch.

Negative log-likelihood. NLL evaluates the probability assigned by the pruned model to the realized dense-generated token y_t:

$$\mathrm{NLL}_t = -\log q_t(y_t). \tag{10}$$

If y_t is sampled from p_t and infinitely many samples are available, its expectation satisfies

$$\mathbb{E}_{y_t \sim p_t}[-\log q_t(y_t)] = H(p_t) + D_{\mathrm{KL}}(p_t \| q_t), \tag{11}$$

where $H(p_t)$ is independent of the pruning candidate. Thus, NLL can be viewed as a sample-based estimate of dense-to-pruned mismatch. However, in practical pruning search, only a limited number of dense-generated trajectories are evaluated. As a result, NLL focuses on the realized token and may ignore the broader next-token distribution, including alternative tokens that are also likely under the dense model and may correspond to valid reasoning continuations.

Entropy. Entropy measures the uncertainty of the pruned model itself:

$$H(q_t) = -\sum_{v \in \mathcal{V}} q_t(v) \log q_t(v). \tag{12}$$

This criterion does not directly compare the pruned model with the dense model. A pruned model can have low entropy while being confidently misaligned with the dense model, or high entropy while still covering dense-preferred tokens. Therefore, entropy captures the sharpness of q_t, but not whether the pruning candidate preserves the dense model's generation behavior.

Jensen–Shannon divergence. JS divergence symmetrizes the distributional mismatch:

$$D_{\mathrm{JS}}(p_t \| q_t) = \frac{1}{2} D_{\mathrm{KL}}(p_t \| m_t) + \frac{1}{2} D_{\mathrm{KL}}(q_t \| m_t), \quad m_t = \frac{1}{2}(p_t + q_t). \tag{13}$$

This makes JS stable and bounded, but also smooths one-sided failures. In particular, when the pruned model assigns insufficient probability to tokens preferred by the dense model, the intermediate distribution m_t partially absorbs this mismatch. Consequently, JS can be less sensitive than forward KL to cases where the pruned model loses probability mass on dense-preferred continuations.

Kullback–Leibler divergence. GDPruner uses the dense-to-pruned KL divergence:

$$D_{\mathrm{KL}}(p_t \| q_t) = \sum_{v \in \mathcal{V}} p_t(v) \log \frac{p_t(v)}{q_t(v)}. \tag{14}$$

This direction is well aligned with preserving the dense model's generation trajectory. It directly penalizes pruning candidates for assigning low probability to tokens that the dense model considers likely.

A simple way to see this is to consider a set of dense-preferred tokens $A \subseteq \mathcal{V}$, such as the top-k tokens under p_t. Let $p_t(A) = \sum_{v \in A} p_t(v)$ and $q_t(A) = \sum_{v \in A} q_t(v)$. By the log-sum inequality, the KL divergence lower-bounds the coarse-grained mismatch over A:

$$D_{\mathrm{KL}}(p_t \| q_t) \geq p_t(A) \log \frac{p_t(A)}{q_t(A)} + (1 - p_t(A)) \log \frac{1 - p_t(A)}{1 - q_t(A)}. \tag{15}$$

Therefore, if pruning causes the model to lose probability mass on the dense-preferred region A, i.e., $q_t(A) \ll p_t(A)$, the KL score increases. This property is desirable for structured pruning because the goal is not merely to produce confident predictions, but to preserve the distributional support that the dense model relies on during autoregressive generation.

Moreover, the dense-to-pruned direction naturally reflects the risk of trajectory deviation. If the pruned model underestimates dense-preferred tokens, subsequent decoding may move to a different prefix, causing errors to accumulate over time. Forward KL is therefore particularly suitable for identifying pruning candidates that preserve stable generation dynamics. This does not imply that KL is universally optimal; other distributional or trajectory-level metrics may capture complementary aspects of generation drift. Nevertheless, among the metrics evaluated in our experiments, dense-to-pruned KL provides the most reliable pruning signal for retaining overall and generative performance.

E Details of Implementations

In this section, we summarize the implementation details of our pruning experiments. For **GDPruner**, we use the generation-drift-guided setting described in the main text and build lightweight self-generated probe trajectories for pruning search. The probe sequence length is set to 128, the tail length is set to 64, and KL divergence is used as the distribution discrepancy metric. For the other four baselines, namely **ShortGPT**, **SLEB**, **LLMDrop**, and **BlockPruner**, we use **WikiText2** [54] as the calibration dataset. Specifically, we sample 128 examples from WikiText2 and set the maximum sequence length to 2048 for all four methods. This unified setting ensures a fair comparison across baseline methods.

For downstream evaluation, we use a zero-shot setting on all benchmarks except GSM8K and Math500. Specifically, GSM8K is evaluated under the 8-shot setting, while Math500 is evaluated under the 4-shot setting. The maximum generation length is set to 256 for generative evaluation, and the random seed is fixed to 42 in all experiments.

For the efficiency analysis, all methods are measured under the same inference setup with a prompt length of 512, a generation length of 128, and a batch size of 1. Latency denotes the end-to-end generation time averaged over 50 runs after one warm-up run, and Tok/s is computed as the number of generated tokens divided by this latency. Speedup is computed as the dense-model latency divided by the pruned-model latency.

For GDPruner, we use $N = 20$ generic seed prompts and top-$k = 128$ for drift estimation. For adaptive search, we set $B_{\min} = 1$, $B_{\mathrm{mid}} = 2$, $B_{\max} = 4$, with thresholds $(\tau_1, \tau_2) = (1.0, 2.0)$ and a sliding-window size of 3.

F Additional Experimental Results

We evaluate Gemma-2-9b-it [55] and Mistral-7B-v0.3 [56] in Table 6. The consistent aggregate gains on Gemma and mixed rankings on Mistral suggest that the benefit of drift-guided selection depends on the backbone and pruning regime. On Mistral, better generative retention does not always coincide with the best overall accuracy, highlighting the need to evaluate generation separately rather than treating aggregate accuracy as a sufficient measure of pruning quality.

Larger models and MoE architectures. Table 7 extends evaluation to Qwen3-32B, Llama-3.1-70B-Instruct, and Mixtral-8×7B-Instruct-v0.1. Owing to their memory requirements, these experiments are conducted on NVIDIA A40 GPUs with 48GB memory. For Mixtral, each complete MoE feed-forward residual sublayer is one pruning unit; expert routing is unchanged. GDPruner's relative advantage in both overall and generative accuracy suggests that generation-drift-based selection remains useful as model scale and feed-forward structure change. Although the margin on Llama-3.1-70B is modest, the consistent direction of the results supports the applicability of the same pruning criterion across the dense and MoE backbones, rather than only within the original small-model setting.

Hyperparameter sensitivity. Table 9 examines probe count N and adaptive-beam thresholds on Llama-3.1-8B-Instruct at 20% pruning, with other settings fixed as in Appendix E. The non-monotonic response to N indicates that simply adding probes does not guarantee better pruning;

Table 6: Performance comparison of structured pruning methods on **Gemma-2-9b-it** and **Mistral-7B-v0.3** under different pruning ratios. AVG denotes the unweighted average over all tasks, and RP denotes the relative performance with respect to the dense model. The best and second-best results within each pruning-ratio group are marked in **bold** and underlined, respectively.

Model	Method	Pruning Unit	Pruning Ratio	Wino.	PIQA	HellaSwag	ARC-e	ARC-c	MMLU	GSM8K	Math500	HumanEval	AVG	RP
Gemma-2-9b-it	Dense	–	0.00%	76.24	80.25	80.01	84.18	65.02	71.92	83.09	39.00	57.32	70.78	100.0
	ShortGPT	Layer	10.00%	**75.45**	79.22	77.60	82.95	62.29	71.44	32.75	20.40	43.29	60.60	85.6
	SLEB	Layer	10.00%	70.72	79.76	77.39	81.57	60.24	64.32	61.64	29.80	**50.00**	63.94	90.3
	LLMDrop	Block	10.00%	72.06	**80.79**	77.38	**83.92**	**63.31**	**71.73**	**79.68**	35.60	34.76	66.58	94.1
	BlockPruner	Block	10.00%	73.56	80.58	79.54	83.04	62.54	67.29	67.93	31.80	48.17	66.05	93.3
	Ours	Block	10.00%	**75.45**	80.20	**79.66**	81.48	61.77	68.38	77.56	**39.00**	43.90	**67.49**	**95.4**
	ShortGPT	Layer	15.00%	70.88	77.69	72.91	81.36	59.56	71.21	6.82	10.40	24.39	52.80	74.6
	SLEB	Layer	15.00%	70.17	79.05	76.96	79.42	59.22	61.62	45.79	24.00	**37.20**	59.27	83.7
	LLMDrop	Block	15.00%	70.88	78.94	73.64	**82.07**	**62.71**	**71.39**	47.16	15.80	4.27	56.32	79.6
	BlockPruner	Block	15.00%	**72.38**	**79.65**	78.80	80.77	58.96	64.12	51.18	29.00	31.10	60.66	85.7
	Ours	Block	15.00%	72.06	79.00	**79.05**	79.88	60.07	63.06	**62.85**	**29.40**	**37.20**	**62.51**	**88.3**
	ShortGPT	Layer	20.00%	70.64	74.05	67.25	75.63	53.67	**71.45**	4.09	7.40	6.10	47.81	67.5
	SLEB	Layer	20.00%	70.80	78.56	73.76	77.78	55.29	58.52	19.26	13.20	**18.29**	51.72	73.1
	LLMDrop	Block	20.00%	69.38	77.31	70.00	77.99	58.19	57.78	1.36	3.40	0.00	46.16	65.2
	BlockPruner	Block	20.00%	68.90	79.16	75.70	78.16	54.01	57.93	19.18	14.40	9.76	50.80	71.8
	Ours	Block	20.00%	**71.98**	**79.49**	**77.80**	**78.62**	**58.45**	62.00	**41.39**	**18.80**	12.80	**55.70**	**78.7**
	ShortGPT	Layer	25.00%	**69.85**	71.98	61.44	70.62	50.17	**69.49**	1.82	3.00	2.44	44.54	62.9
	SLEB	Layer	25.00%	65.43	77.58	68.26	73.48	48.72	47.92	7.20	6.40	9.76	44.97	63.5
	LLMDrop	Block	25.00%	68.75	74.43	66.12	70.58	52.30	48.23	1.21	2.20	0.00	42.65	60.3
	BlockPruner	Block	25.00%	68.43	**78.51**	**74.56**	**77.02**	**53.41**	58.20	17.06	11.80	8.54	49.73	70.3
	Ours	Block	25.00%	69.38	77.48	74.55	74.92	51.71	53.30	**22.44**	**12.40**	**12.20**	**49.82**	**70.4**
Mistral-7B-v0.3	Dense	–	0.00%	74.19	81.99	80.72	80.18	54.35	59.66	37.53	11.60	0.61	53.43	100.0
	ShortGPT	Layer	10.00%	73.48	78.56	77.04	73.70	49.74	58.76	19.41	6.00	**1.22**	48.66	91.1
	SLEB	Layer	10.00%	65.67	79.82	74.01	73.19	44.62	36.52	2.73	5.20	0.00	42.42	79.4
	LLMDrop	Block	10.00%	**73.88**	81.28	80.16	**80.09**	**54.01**	**59.76**	**34.95**	9.20	0.00	52.59	98.4
	BlockPruner	Block	10.00%	73.64	81.61	79.01	78.70	51.28	57.68	24.72	8.60	0.00	50.58	94.7
	Ours	Block	10.00%	**73.88**	**82.37**	**80.54**	80.05	53.16	59.46	33.89	**10.60**	**1.22**	**52.80**	**98.8**
	ShortGPT	Layer	15.00%	72.53	76.17	73.01	67.85	44.97	57.35	4.93	4.80	**2.44**	44.89	84.0
	SLEB	Layer	15.00%	63.54	78.07	68.35	68.52	40.44	28.06	2.65	3.40	0.00	39.23	73.4
	LLMDrop	Block	15.00%	**73.48**	**80.74**	**79.66**	**79.29**	**53.33**	**59.66**	14.63	3.80	1.83	**49.60**	**92.8**
	BlockPruner	Block	15.00%	73.01	80.47	76.04	77.65	49.23	52.42	14.63	5.20	0.00	47.63	89.1
	Ours	Block	15.00%	72.30	79.98	76.88	75.51	48.81	58.76	**16.07**	**6.20**	0.61	48.35	90.5
	ShortGPT	Layer	20.00%	71.90	73.34	70.02	64.06	42.92	**58.08**	1.82	2.80	**0.61**	42.84	80.2
	SLEB	Layer	20.00%	63.46	76.71	66.64	65.03	37.97	28.06	2.12	3.60	0.00	38.18	71.5
	LLMDrop	Block	20.00%	**72.85**	77.69	**75.49**	69.91	**46.67**	51.32	1.06	2.00	0.00	**44.11**	**82.6**
	BlockPruner	Block	20.00%	68.35	**78.94**	72.82	**74.28**	43.60	46.25	**5.69**	3.40	0.00	43.70	81.8
	Ours	Block	20.00%	68.67	78.56	74.10	70.88	41.81	49.28	4.32	**5.20**	**0.61**	43.71	81.8
	ShortGPT	Layer	25.00%	66.06	67.63	54.46	49.45	36.95	**50.85**	0.61	1.00	0.00	36.33	68.0
	SLEB	Layer	25.00%	61.09	73.61	62.79	60.44	35.49	27.52	2.27	3.00	0.00	36.25	67.8
	LLMDrop	Block	25.00%	**70.24**	72.80	69.46	60.52	39.93	24.73	0.45	1.40	0.00	37.73	70.6
	BlockPruner	Block	25.00%	67.09	76.93	69.59	**69.87**	38.57	36.20	2.65	2.40	0.00	40.37	75.6
	Ours	Block	25.00%	67.48	**77.53**	**73.70**	69.19	**41.13**	36.83	**2.97**	**3.40**	**0.61**	**41.43**	**77.5**

a smaller probe set can approach the default overall accuracy, although generative retention is more sensitive. Across the tested threshold pairs, overall accuracy remains comparatively stable despite variation in generative performance. Together, these observations support effective drift-guided pruning with a compact probe budget and flexibility in beam-threshold selection within our experimental setting.

G Post-Pruning Recovery

LLM pruning aims to produce a sparse model, while performance restoration seeks to reduce the gap between the pruned and dense models. Although GDPruner preserves performance better than existing pruning methods, model quality still declines as the pruning ratio increases, indicating that aggressive pruning inevitably removes part of the model's useful capacity. To further mitigate this degradation, we combine GDPruner with Low-Rank Adaptation (LoRA) [57] and Quantized Low-Rank Adaptation (QLoRA) [58] as lightweight post-pruning restoration strategies. Both adapters are trained on WikiText2 using 2,000 training samples and 256 validation samples, with a maximum sequence length of 256. The LoRA rank and α are set to 8 and 16, respectively, with dropout 0.05, a learning rate of 2×10^{-4}, batch size 2, and gradient accumulation steps of 8.

Fig. 8 reports the post-pruning restoration results. Lightweight adaptation generally improves GDPruner-pruned models across both architectures, indicating that the removed-block structure produced by GDPruner remains compatible with parameter-efficient recovery. The gains are more evident at medium and high pruning ratios, where the pruned model has lost more capacity and thus benefits more from task-agnostic adaptation. This suggests that GDPruner does not over-prune fragile

Table 7: Performance comparison on additional LLMs without recovery. AVG and Gen. Acc. denote the unweighted averages (%) over all nine tasks and over GSM8K, Math500, and HumanEval, respectively. The best and second-best results within each model–ratio group are highlighted in **bold** and underlined.

Model	Method	Pruning Ratio	AVG ↑	Gen. Acc. ↑
Mixtral-8×7B -Instruct-v0.1	ShortGPT	20%	48.28	2.63
	LLMDrop	20%	42.74	0.15
	SLEB	20%	49.30	6.35
	BlockPruner	20%	<u>49.65</u>	<u>8.21</u>
	Ours	20%	**50.92**	**10.16**
Qwen3-32B	ShortGPT	25%	<u>60.35</u>	<u>46.59</u>
	LLMDrop	25%	40.87	9.19
	SLEB	25%	58.24	43.56
	BlockPruner	25%	58.01	40.08
	Ours	25%	**61.75**	**48.75**
Llama-3.1-70B -Instruct	ShortGPT	25%	61.87	35.34
	LLMDrop	25%	58.59	19.06
	SLEB	25%	64.31	43.33
	BlockPruner	25%	<u>68.63</u>	<u>49.79</u>
	Ours	25%	**68.90**	**50.61**

Table 8: Post-pruning results with matched LoRA recovery on Qwen3-32B at 25% pruning.

Method + LoRA	AVG ↑	Gen. Acc. ↑
ShortGPT	<u>65.76</u>	<u>52.55</u>
LLMDrop	59.62	35.06
SLEB	63.39	48.36
BlockPruner	64.46	49.51
Ours	**67.11**	**55.72**

Table 9: Sensitivity to probe count and adaptive-beam thresholds on Llama-3.1-8B-Instruct at 20% pruning.

Parameter	Value	AVG ↑	Gen. Acc. ↑
N	5	50.28	14.04
	10	47.63	8.47
	20	50.93	19.24
	40	47.24	11.61
(τ_1, τ_2)	$(1.0, 2.0)$	50.93	19.24
	$(1.5, 3.0)$	50.86	22.15
	$(2.0, 4.0)$	49.92	18.49

components in a way that makes subsequent recovery difficult; instead, it preserves a sparse backbone that can still be effectively adapted by LoRA or QLoRA, consistent with the view that solutions lying in flatter regions of the loss landscape generalize more robustly under perturbation [59]. Overall, these results show that GDPruner is complementary to post-pruning restoration and can serve as a strong starting point for further lightweight recovery.

Matched recovery. We apply the same LoRA recovery protocol to all five pruning methods on Qwen3-32B at 25% pruning (Table 8). GDPruner retains the strongest overall and generative accuracy after recovery, suggesting that effective block selection remains important even when subsequent adaptation is available. These results support generation-drift-guided pruning as a strong starting point for lightweight recovery, extending its benefits beyond training-free compression.

H Time Costs of Pruning Methods

Table 10 reports offline pruning costs at 25% pruning on RTX A5000 and A40 GPUs. Calibration-based methods use 128 WikiText2 examples of at most 2,048 tokens, following Appendix E. Each run uses one GPU; the hardware groups represent separate measurements rather than a controlled comparison of GPU speed.

Although one-shot ranking is faster, GDPruner evaluates block interactions at substantially lower offline cost than SLEB and BlockPruner in the tested settings. Compact self-generated probes and cached dense outputs enable subset-level search without repeated corpus-level PPL evaluation.

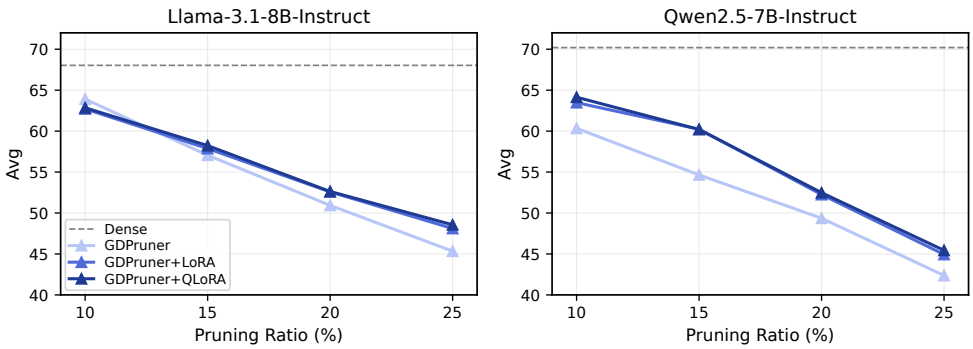


Figure 8: Post-pruning restoration results of GDPruner with LoRA and QLoRA on Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct across pruning ratios.

Table 10: Execution time of different pruning methods at 25% pruning on one NVIDIA GPU. Times are rounded to whole minutes.

Model	GPU	Ours	ShortGPT	SLEB	LLMDrop	BlockPruner
Llama-3.1-8B-Instruct	A5000	13 mins	<1 min	33 mins	<1 min	2 hours 2 mins
Qwen2.5-7B-Instruct		11 mins	<1 min	33 mins	<1 min	2 hours 6 mins
Mistral-7B-v0.3		13 mins	<1 min	31 mins	<1 min	2 hours 1 min
Qwen2.5-0.5B	A40	1 min	<1 min	8 mins	<1 min	32 mins
Qwen2.5-1.5B		2 mins	<1 min	13 mins	<1 min	56 mins
Qwen2.5-3B		6 mins	<1 min	30 mins	<1 min	2 hours

Together with the generative retention gains reported above, this provides a favorable balance between pruning quality and search cost, with the additional search performed only once before deployment.

I Text Generation Results on Pruned Models

We provide qualitative examples comparing the dense model, BlockPruner, and GDPruner across open-ended writing, social reasoning, mathematical reasoning, and code generation. As shown in Tables 11–13, BlockPruner more often exhibits repetition, semantic degeneration, unstable reasoning, or syntactically invalid code, whereas GDPruner better preserves coherent topic development and logical progression. These examples provide intuitive evidence that generation-drift-guided pruning better maintains the stability required for long-horizon autoregressive reasoning and structured generation.

Table 11: Generated examples on open-ended generation for Llama-3.1-8B-Instruct. The underlined text denotes the input prompt.

Method	Generated Sentences
Prompt 1 (Domain: Philosophy / Open-ended Generation). To be or not to be, that is the question that has been asked thousands of times throughout history. But have you ever wondered what it would be like to not exist?	
Dense	The concept of non-existence is a complex and abstract idea that has puzzled philosophers, scientists, and thinkers for centuries. It's a topic that raises many questions and sparks interesting discussions. Let's dive into the idea of non-existence and explore what it might be like.
BlockPruner	If you didn't exist, there would be one less person on this earth. You would not be able to experience the world around you, and the world would be a very different place without you. You wouldn't be able to experience the beauty of a sunset, the joy of a warm summer day, or the beauty of a beautiful sunset.
Ours	You're referring to the philosophical concept of non-existence. It's a mind-boggling thought experiment that has puzzled thinkers for centuries. If we're considering the possibility of a non-existent existence, we're essentially asking ourselves: what is the nature of existence and non-existence.
Prompt 2 (Domain: Work / Social Reflection). What are the hidden challenges of working from home that people do not notice at first?	
Dense	While working from home can be convenient and comfortable, there are several hidden challenges that people may not notice at first. Here are some of the common ones: social isolation, blurred boundaries between work and personal life, distractions, lack of structure, and limited resources.
BlockPruner	When people start working from home, they often think it's a dream come true. But one of the biggest challenges is the lack of structure. Without a traditional office environment, it's easy to get sidetracked by household chores, family members, and personal errands. It's easy to get sucked into the many distractions of home life, and it can be hard to separate work and personal life.
Ours	There are several hidden challenges of working from home that people may not notice at first. These include blurred lines between work and personal life, social isolation, lack of clear boundaries, and distractions from family members, pets, or household chores, all of which can gradually reduce focus and well-being.
Prompt 3 (Domain: Reflective Writing / Personal Growth). Write a reflective paragraph on what it means to become a better listener.	
Dense	Becoming a better listener is a transformative journey that requires a profound shift in perspective and approach. It's not just about hearing the words being spoken, but about truly understanding the emotions, thoughts, and experiences behind them. When we become better listeners, we open ourselves up to a deeper connection with others.
BlockPruner	To become a better listener, one must cultivate a mindset that values and respects the thoughts and perspectives of others. This involves being fully present in conversations, maintaining eye contact, maintaining eye contact, maintaining eye contact, maintaining eye contact, maintaining eye contact.
Ours	Becoming a better listener is a skill that requires a combination of self-awareness, active listening, and a willingness to understand different perspectives. It's not just about hearing the words, but also about understanding the emotions and emotions behind them. When we become better listeners, we are able to create a safe space for others to express themselves.

Table 12: Generated examples on law, ethics, and future scenarios for Llama-3.1-8B-Instruct. The underlined text denotes the input prompt.

Method	Generated Sentences
Prompt 4 (Domain: Social Psychology / Authenticity). Why do people admire authenticity, and how can it be faked?	
Dense	Authenticity is a highly valued trait in many cultures, and people often admire it for several reasons. Authentic individuals are perceived as genuine, transparent, and honest, which builds trust with others. They are often seen as more relatable and down-to-earth.
BlockPruner	The concept of authenticity is often associated with authenticity, authenticity is a quality that is often associated with authenticity, and it is often associated with authenticity. Authenticity is often associated with authenticity, and it is often associated with authenticity.
Ours	The desire for authenticity is a complex and multifaceted aspect of human nature. People often value authenticity because it represents a genuine and sincere approach to life. People are drawn to authenticity because it builds trust and credibility. However, it's not always easy to distinguish between authenticity and fake authenticity.
Prompt 5 (Domain: Economics / Social Reasoning). Discuss whether economic growth always improves quality of life.	
Dense	Economic growth is often associated with improvements in quality of life, but it's not a straightforward relationship. While economic growth can bring many benefits, it's not a guarantee that quality of life will improve for everyone. Here are some arguments for and against the idea that economic growth always improves quality of life.
BlockPruner	No, not always. While economic growth can bring about improvements in living standards, it's not a guarantee of a better life. There are instances where economic growth can have negative consequences, such as environmental degradation, income inequality, and social inequality. The widening of the wealth gap can lead to social unrest and decreased social cohesion.
Ours	Economic growth is often associated with improvements in quality of life, but it's not a guarantee that quality of life will improve for everyone. Here are some arguments for and against the idea that economic growth always improves quality of life. Increased access to goods and services, job creation, and improved healthcare can help, but not all benefits are distributed equally.
Prompt 6 (Domain: Education / AI Futures). Describe a future classroom in which humans and AI collaborate naturally.	
Dense	In the year 2050, the traditional classroom has undergone a revolutionary transformation. The once static and hierarchical learning environment has given way to a dynamic, collaborative space where humans and artificial intelligence (AI) work together seamlessly. This futuristic classroom is designed to foster a symbiotic relationship between students, teachers, and AI.
BlockPruner	The future of education is likely to be a collaborative effort between humans and AI, with the integration of AI in the classroom becoming the new norm. Here's a possible vision of the future of education with the integration of AI. The system can provide real-time feedback, provide real-time feedback to the teacher, and provide real-time feedback to the students.
Ours	In the year 2050, the classroom of the future is a place where humans and artificial intelligence (AI) work together in perfect harmony. The room is a spacious, well-equipped learning environment that fosters creativity, critical thinking, and innovation. The walls are adorned with a large, wrap-around display screen that serves as a virtual whiteboard.

Table 13: Generated examples on mathematical reasoning and code generation for Llama-3.1-8B-Instruct. The underlined text denotes the input prompt. For the code example, only the leading correct segment of the generated code is shown for readability.

Method	Generated Sentences / Code
Prompt 7 (Domain: Mathematical Reasoning). A merchant buys a product for 80 dollars, marks it up by 25%, and then gives a 10% discount on the marked price. What is the final selling price? Show the calculation clearly.	
Dense	To find the final selling price, we first compute the marked-up price: $80+0.25\times80 = 100$. Then we apply the 10% discount: $100 - 0.10 \times 100 = 90$. Therefore, the final selling price is 90.
BlockPruner	To find the final price, we first calculate the markup and then apply the discount. However, the reasoning drifts into using the wrong base price and concludes with repeated statements such as "The final answer is 92.5," which is incorrect.
Ours	The initial price is 80. After a 25% markup, the price becomes $80 + 20 = 100$. Then a 10% discount is applied to 100, which is 10 dollars. Therefore, the final selling price is $100 - 10 = 90$.
Prompt 8 (Domain: Code Generation). Write a Python function `remove_duplicates(nums)` that preserves the original order while removing duplicate elements from a list.	

Dense	`def remove_duplicates(nums):`
	`  seen = set()`
	`  return [x for x in nums if not (x in seen or seen.add(x))]`

BlockPruner	`def remove_duplicates(nums):`
	`  return [x for n in set() from functools.reduce(`
	`  lambda s, x in nums: ...`

Ours	`def remove_duplicates(nums):`
	`  seen = []`
	`  result = []`
	`  for i, item in enumerate(nums):`
	`  if item not in seen:`
	`  seen.append(item)`
	`  result.append(item)`
	`  return result`

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The Abstract and Introduction summarize the paper's main contributions and clearly delimit its scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Sec. 6 discusses the main limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not include formal theoretical claims. The analysis is used to provide empirical evidence and intuitive motivation for pruning.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Please refer to Sec. 4 and Appendix E.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Our source code is included in the supplementary material and will be publicly released upon publication.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Sec. 4 and Appendix E provide the detailed training settings including hyperparameters.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper does not include error bars because of limited computational resources that prevented extensive experimentation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Please see details of Experimental Setup in Sec. 4.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and ensure that our paper strictly adheres to these guidelines.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss the broader impacts in Sec. 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risk as we only used existing publicly available assets.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All of these are properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: Related codes will be released with suitable license.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve human subjects, crowdsourcing, or participant data collection.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs were used only for writing and editing assistance.

Guidelines:

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