POP: Online Structural Pruning Enables Efficient Inference of Large Foundation Models

Yi Chen[1] **Wonjin Shin**[1] **Shuhong Liu**[2] **Tho Mai**[1] **Jeongmo Lee**[1]
Chuanbo Hua[1] **Jun Liu**[3] **Kun Wang**[1] **Joo-Young Kim**[1]
[1]KAIST, Daejeon, South Korea [2]University of Tokyo, Tokyo, Japan
[3]Tokyo Institute of Technology, Tokyo, Japan
{chenyi,2swj1202,thomh1511,dlsxprmfkf1234,cbhua,walkerwang,jooyoung1203}@kaist.ac.kr
s-liu@mi.t.u-tokyo.ac.jp liujun.devops@gmail.com

Abstract

Large foundation models (LFMs) achieve strong performance through scaling, yet current structural pruning methods derive fixed pruning decisions during inference, overlooking sparsity patterns that emerge in autoregressive token generation. We propose POP (**P**artition-guided **O**nline **P**runing), an efficient online structural pruning framework that enables context-conditioned dynamic pruning with low, tunable computational overhead. POP partitions model channels into retained, candidate, and pruned regions, where prefilling defines a batch-shared coarse pruning partition, and decoding generates a per-sample fine-grained mask within the candidate region, avoiding full-channel re-evaluation. The coarse pruning partition preserves consistently important weights, while the fine-grained masking provides context-conditioned variation during decoding. Moreover, POP is a lightweight, plug-and-play method that requires no offline calibration, retraining, or learned predictors. Extensive evaluations across diverse LFMs, including large language models (LLMs), mixture-of-experts models (MoEs), and vision–language models (VLMs), demonstrate that POP achieves competitive accuracy on short-form QA tasks and strong generation performance relative to existing structural pruning baselines, while maintaining efficient inference. Our code is available at https://github.com/Yi1-Chen/POP.

1 Introduction

Large foundation models (LFMs) [8], such as large language models (LLMs) [9, 11], vision–language models (VLMs) [65, 2], and mixture-of-experts models (MoE) [18, 20, 49], have achieved remarkable performance through continuous scaling [35]. However, this scaling trend incurs substantial computational overhead, posing a major challenge for efficient inference. To mitigate this, various compression techniques [26], such as quantization [33, 16], low-rank decomposition [15, 30], and pruning [27, 26, 70], have been extensively studied. Among them, structural pruning [3, 44, 56, 41] is particularly appealing, as it removes entire computational structures (e.g., weight channels) without relying on specialized hardware support [75, 76, 31].

Early structural pruning methods for LLMs typically derive a static pruning mask through offline calibration [56, 3, 44] and apply it uniformly throughout inference. Although effective at reducing model size and computational cost, such static strategies are input-agnostic and fail to capture the diverse sparsity patterns induced by different contexts and tasks. Recent studies have therefore explored context-conditioned [29, 41] or dynamic pruning [50] approaches that adapt pruning decisions across input prompts, often without retraining.

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

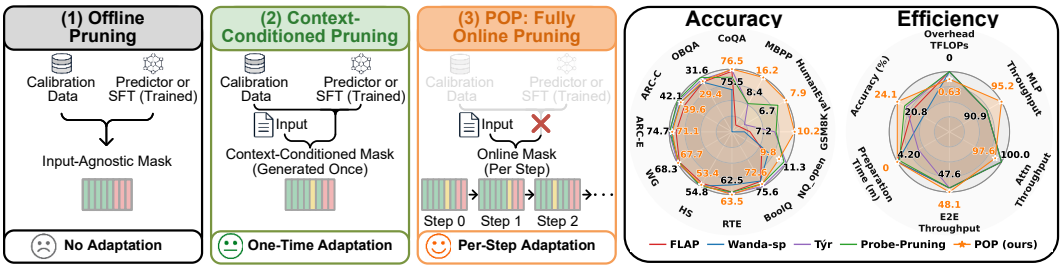

Figure 1: Overview of pruning paradigms and performance comparison. **Left:** Three pruning paradigms with increasing levels of online adaptation. (1) Offline pruning generates a static, input-agnostic mask. (2) Context-conditioned pruning adapts the mask once based on the input prompt. (3) POP dynamically generates candidate masks at every decoding step. **Right:** Task-wise performance across benchmarks, together with inference efficiency on Llama2-7B. POP exhibits better performance across tasks than baseline pruning methods while maintaining inference efficiency.

Despite this progress, online pruning remains challenging. Most existing approaches make pruning decisions only once during the prefill stage and apply the resulting decision throughout autoregressive generation. Such static pruning strategies fail to adapt to evolving decoding contexts, which can conflict with the inherently input-dependent computation behavior of LLMs and result in substantial performance degradation on generation tasks compared to dense models. This issue is reflected in our empirical observations. For example, under a 20% pruning ratio, the model pruned by the static framework Týr [44] retains 98% accuracy on short-form QA tasks such as ARC-E [13], with LLaMA2-7B [72], but preserves only 35% performance on long-form generation benchmarks such as MBPP [5]. These observations expose the limitations of static pruning. These results suggest that pruning decisions benefit from generation-context conditioning during online inference.

A natural solution is to update pruning decisions dynamically during decoding. However, most existing context-conditioned pruning methods target multi-token settings. For example, Probe Pruning [41] estimates importance via probing, while Instruction-Following Pruning [29] uses an input-conditioned predictor; both require activations aggregated over multiple tokens. This makes them fundamentally incompatible with autoregressive decoding, where only one token is available per step. Even if one applies probing-based importance estimation to the generated token itself, it would further introduce an additional forward pass, incurring prohibitive runtime overhead. These limitations motivate an online pruning framework that supports context-conditioned adaptation with low, tunable overhead and broad applicability across tasks and model architectures.

To narrow the gap between pruned and dense models while keeping runtime overhead low, we propose POP (**P**artition-guided **O**nline **P**runing), an online structural pruning framework for context-conditioned pruning during autoregressive inference with low, tunable computational overhead. POP is lightweight and plug-and-play, requiring no offline calibration, retraining, or predictor learning, and can be readily applied to diverse LFMs. The main contributions are summarized as follows:

- **We demonstrate the benefits of online pruning during autoregressive generation.** We show that pruning decisions fixed during prefilling can become less suitable for newly generated tokens because they remain determined by the initial prompt, highlighting the benefit of online, dynamic pruning for addressing contextual sparsity during decoding.
- **We propose POP: a Partition-guided Online Pruning framework that balances accuracy and efficiency.** POP introduces a tri-state channel partition based on prefill-stage importance, dividing channels into retained, pruned, and candidate regions, where only candidate channels are dynamically selected during decoding, enabling fine-grained context adaptivity with low, tunable overhead.
- **We develop POP as a deployment-friendly framework for diverse LFMs.** Without offline calibration, auxiliary predictors, or retraining, POP can be seamlessly applied to a wide range of LFMs, enabling practical, scalable inference-time acceleration.

We empirically evaluate POP on diverse LFMs, including representative LLMs, MoE models, and VLMs, and observe competitive QA accuracy and strong generation performance against state-of-the-art structural pruning methods. In particular, POP improves Llama3.1-8B [19] by 5.27%

Table 1: Comparison of pruning methods. Our approach prunes based on contextual input at inference time, requiring neither calibration nor additional training overhead. *SFT: Supervised Fine-Tuning.* Structural reduction refers to removing entire channels or modules.

Property	LLM-P [56]	S-GPT [21]	Slice [4]	W-sp [3]	FLAP [3]	Týr [44]	DejaVu [53]	IF-P [29]	Skip [89]	μ-MoE [38]	PP [41]	POP
Structural reduction	✓	×	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
No SFT	×	✓	✓	✓	✓	✓	✓	×	×	✓	✓	✓
No calibration	×	×	×	×	×	×	×	×	✓	✓	×	✓
No predictor	×	✓	✓	✓	✓	✓	×	×	×	✓	✓	✓
Context-conditioned	×	×	×	×	×	×	✓	✓	✓	✓	✓	✓

on short-form QA and 17.71% on generation compared to the SOTA online pruning method [41]. Moreover, with only 4.34% relative FFN FLOPs overhead, POP achieves a $1.29\times$ FFN-layer speedup and a $1.14\times$ end-to-end speedup on Llama2-7B.

2 Related Work

2.1 Contextual Sparsity in LLMs

LLMs exhibit contextual sparsity, where only a subset of neurons is activated for a given input [60, 24, 7, 55]. This motivates conditional execution via learned predictors or adaptive thresholds. Deja Vu [53] selects attention heads and FFN neurons using per-layer predictors, with follow-ups improving robustness [1]. Training-free methods suppress low-utility activations or apply runtime thresholds [42, 91, 51]. System studies show power-law neuron activation, with consistently activated *hot* neurons and context-dependent *cold* neurons [69], while ReLU-based LLMs exhibit *aggregated sparsity* during autoregressive generation [62]. These methods are related to POP but target a different trade-off: token-level activation sparsity and thresholding versus bounded online structural pruning at the FFN channel granularity, which yields speedup through reduced dense-matmul shapes. We discuss this distinction in Appendix A.5.

MoE models realize contextual sparsity at a coarser granularity by replacing dense FFNs with multiple experts and routing each token to a small subset [43, 92], allowing parameter scaling while keeping per-token FLOPs bounded by the number of activated experts [17].

2.2 Unstructured & Structural Pruning

Pruning compresses LLMs by removing parameters while preserving accuracy. Unstructured methods achieve high sparsity with limited recovery cost: SparseGPT [21] uses one-shot reconstruction, and Wanda [70] leverages activation-based saliency, with later work improving sparsity allocation [78, 82]. However, irregular sparsity often yields limited speedup on GPUs due to poor compatibility with dense kernels. Structured pruning removes entire channels or modules for more hardware-friendly acceleration [75, 76, 31]. LLM-Pruner [56] prunes coupled substructures, while LoRAPrune [87] integrates pruning with low-rank adaptation [30]. SliceGPT [4] and ZipLM [39] further align pruning with dense inference efficiency, and Sheared-LLaMA [77] combines pruning with continued training. Recent work explores adaptive sparsity across layers or blocks [90, 88, 58, 81, 28], including fluctuation-based criteria (FLAP [3]) and global optimization via evolutionary search [44].

2.3 Context-Conditioned Pruning

Context-conditioned pruning adapts masks per input instead of using a static post-training mask. EBERT [52] introduces input-conditioned structured pruning via lightweight predictors. Recent work extends this paradigm with richer conditioning and objectives: Instruction-Following Pruning [29] learns instruction-aware masks, and Probe Pruning [41] performs batch-specific structured pruning via lightweight probing. Alignment-aware constraints improve robustness under distribution shift [63]. RAP [50] formulates runtime pruning as a control problem with RL under memory budgets, while μ-MoE [38] treats prompt-adaptive weight pruning as micro-grained expert selection. POP instead combines shared channel removal with per-step candidate masks.

3 Method

The objective of POP is to offer online structural pruning at inference time to improve computational efficiency and minimize latency while preserving accuracy. The comparisons between POP and previous works are listed in Table 1. We begin with the preliminaries of structural pruning and LLM architectures in Section 3.1. Motivated by the contextual sparsity of LFMs, we empirically show the correlation of pruning partitions and formulate POP in Section 3.2. Section 3.3 subsequently elaborates the two-stage architecture. Section 3.4 describes our pruning metric and computation overhead.

3.1 Preliminaries and Notations

Notation. We denote a linear transformation by a weight matrix $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$, where C_{in} and C_{out} indicate the input and output channel dimensions, respectively. Let $\mathbf{X} \in \mathbb{R}^{B \times L \times C_{\text{in}}}$ denote the input activations, where B is the batch size and L is the sequence length. During autoregressive decoding, the activation at step t is denoted as $\mathbf{X}_t \in \mathbb{R}^{B \times 1 \times C_{\text{in}}}$. We use i to index the structural channel being pruned, and k to index input channels of a weight matrix. Channel-wise importance scores are denoted by I_i, and their step-wise counterparts during decoding are written as $I_i(t)$. The FFN channel-mask ratio is denoted by r, whereas PR denotes the model-level parameter-reduction ratio reported in experiments.

Weight-Activation Structural Pruning. We consider the common activation-aware formulation for structural pruning, where importance is estimated jointly from model weights and input activations [70]. The element-wise importance weight can be defined as

$$I_{i,k} = |W_{i,k}| \cdot \|\mathbf{X}_{:,:,k}\|_2, \tag{1}$$

where $\|\mathbf{X}_{:,:,k}\|_2$ denotes the ℓ_2 norm of the k-th input channel activation across tokens and samples.

While importance can be computed for individual weights, structural pruning operates on channel-level structures rather than isolated parameters. Accordingly, following [3], element-wise importance scores are aggregated to obtain output-channel importance, which can be defined as

$$I_i = \mathcal{A}\left(\{I_{i,k}\}_{k=1}^{C_{\text{in}}}\right), \quad i = 1, \ldots, C_{\text{out}}, \tag{2}$$

where $\mathcal{A}(\cdot)$ is an aggregation operator mapping element-wise scores to channel-level importance.

The resulting channel-importance vector $\mathbf{I}$ provides the pruning criterion, with decisions applied to entire channels. For feed-forward networks (FFNs), this structural channel is the intermediate dimension shared by consecutive projections, i.e., the output of the up/gate projection and the input of the final projection, with dimension C_{int}.

3.2 Contextual Sparsity for Online Pruning

As discussed in the preliminaries, importance estimation in LFMs depends jointly on fixed model weights and input-dependent activation statistics. During inference, this coupling induces a structural mismatch: pruning targets remain fixed at the channel level, whereas the activation patterns used for importance estimation vary across inference stages. In particular, while prefilling computes importance from full-sequence activations of the input prompt, decoding relies on step-wise activations $\mathbf{X}_t$, in which the activation sequence used for scoring shrinks from length L to a single token, leading to token-dependent fluctuations in channel importance.

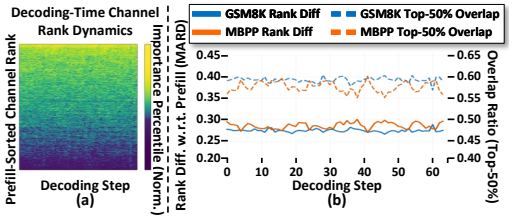

Figure 2: Decoding-time structural dynamics under online pruning. (a) Channel-rank evolution across decoding steps. (b) Mean rank difference and Top-50% channel overlap on GSM8K and MBPP using Llama3.1-8B, reflecting ranking variation during decoding. Additional ranking analyses across models, tasks, and layers are provided in Appendix B.3.

Figure 2 provides a direct visualization of this phenomenon. Channels are sorted according to their importance ranking obtained at the prefill stage, and their relative rankings at each decoding step

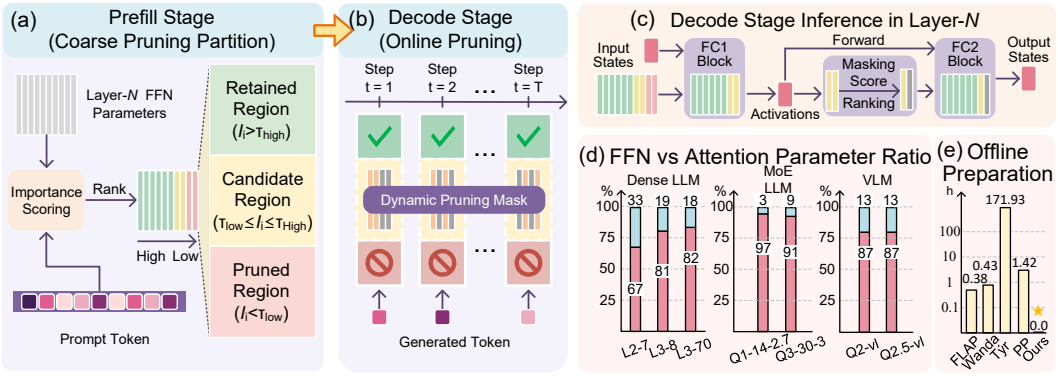

Figure 3: The overview of POP. (a) During the prefill stage, FFN channels in each layer are ranked by importance and partitioned into retained, candidate, and pruned regions, forming a coarse pruning partition. (b) During decoding, POP performs online pruning by dynamically selecting candidate channels at each generation step using step-wise pruning masks. (c) At each decoding step, intermediate activations are used to compute importance scores of candidates and update the pruning mask before the final FFN computation in layer N. (d) The proportion of FFN parameters dominates over attention parameters across dense LLMs, MoE LLMs, and VLMs, motivating our FFN-focused pruning. (e) Compared with prior methods, POP requires no offline preparation cost.

are tracked thereafter. As illustrated in Figure 2(a), channel rankings exhibit noticeable temporal variations during decoding, while the overlap of high-importance channels with the prefilling reference exceeds the random baseline. This behavior is further quantified in Figure 2(b). The mean rank difference captures deviations from the prefill ranking, while Top-50% overlap measures the fraction of channels from the prefill top half that remain in the top half during decoding. Across both GSM8K and MBPP, the overlap remains above the 50% random baseline despite persistent ranking fluctuations.

Together, these observations reveal a characteristic pattern of decoding-time behavior: the importance structure remains partially stable, whereas intermediate channels exhibit context-dependent fluctuations. We refer to this coexistence of structural stability and localized variation as *contextual sparsity*. As a result, pruning decisions determined solely at the prefill stage are insufficient to fully capture decoding-time computation, motivating online structural pruning that adapts pruning decisions to the evolving generation context.

3.3 POP Architecture

Motivated by the contextual sparsity observed above, we design POP as a two-stage online pruning framework that separates coarse-grained pruning decisions from fine-grained, context-conditioned pruning during autoregressive decoding. The overview of POP is illustrated in Figure 3 (a) and (b). We focus on pruning FFNs, which dominate the model size as shown in Figure 3 (d), while the proposed framework can be extended to other components such as attention heads.

Calibration Stage. Unlike previous approaches [70, 3, 41, 44] that require offline calibration, shown in Figure 3 (e), POP is a plug-and-play framework that requires no offline preparation or training, and is ready for out-of-the-box inference.

Prefill Stage. Inference begins with the *prefill* stage, which processes all prompt tokens in parallel and produces full-sequence activations $\mathbf{X} \in \mathbb{R}^{B\times L\times C_{\text{in}}}$. These activations offer a global view of channel utilization under the given input prompt. Leveraging this information, POP derives an initial *coarse pruning partition*. Channel-wise importance scores I_i are computed from the intermediate activations $\mathbf{X}^{\text{int}}$ obtained by passing the input prompt through the up/gate projection. Scores aggregate over prompt tokens and samples, producing a batch-shared partition.

Given a target pruning ratio r and a partition fraction γ, POP defines an importance band around the pruning threshold. Let q_α denote the α-quantile of importance scores $\{I_i\}_{i=1}^{C_{\text{int}}}$. Two thresholds are defined as $\tau_{\text{low}} = q_{r-\gamma}$ and $\tau_{\text{high}} = q_{r+\gamma}$. Channels are partitioned into three disjoint regions:

Retained Region ($\mathcal{R}$): $\mathcal{R} = \{\, i \mid I_i > \tau_{\text{high}} \,\}$. Channels in this region exhibit consistently high importance and are always preserved, forming a stable computational backbone throughout inference.

Pruned Region ($\mathcal{P}$): $\mathcal{P} = \{\, i \mid I_i < \tau_{\text{low}} \,\}$. These channels contribute marginally and are removed upfront to improve inference efficiency for the given input prompt.

Candidate Region ($\mathcal{C}$): $\mathcal{C} = \{\, i \mid \tau_{\text{low}} \leq I_i \leq \tau_{\text{high}} \,\}$. This region captures channels with context-conditioned variability and defines a bounded *dynamic search space* for online pruning.

This tri-state partition is conceptually related to the hot/cold neuron categorization in the activation sparsity literature [69, 47], where hot neurons map to $\mathcal{R}$ and cold neurons span $\mathcal{C} \cup \mathcal{P}$. However, unlike activation gating which preserves full model capacity, our partition drives irreversible structural removal: channels in $\mathcal{P}$ are permanently excluded from computation for a given input, achieving genuine FLOPs reduction without custom sparse kernels.

Algorithm 1: POP: Online Structural Pruning

Input: Prompt $\mathbf{X}$, pruning ratio r, partition fraction γ
Prefill Stage:
Compute channel importance I_i from $\mathbf{X}^{\text{int}}$ `// Eq. 3`
$\tau_{\text{low}} \leftarrow q_{r-\gamma}$, $\tau_{\text{high}} \leftarrow q_{r+\gamma}$ `// Thresholds`
$\mathcal{R}, \mathcal{P}, \mathcal{C} \leftarrow \text{Partition}(\{I_i\}, \tau_{\text{low}}, \tau_{\text{high}})$ `// Tri-state`
Complete prefilling with channels $\mathcal{R} \cup \mathcal{C}$
Decode Stage:
for *each decoding step* $t = 1, 2, \ldots$ **do**
  $\mathbf{H}_t \leftarrow \text{Forward}(\mathbf{X}_t;\ \mathcal{R} \cup \mathcal{C})$ `// Partial fwd`
  Evaluate $I_i(t)$ from $\mathbf{H}_t$ for $i \in \mathcal{C}$
  $\mathcal{C}_t \leftarrow \text{Select}(\{I_i(t)\}_{i \in \mathcal{C}}, r)$
  Output $\leftarrow \text{Project}(\mathbf{H}_t;\ \mathcal{R} \cup \mathcal{C}_t)$
end

Decode Stage. Following the prefill stage, autoregressive models proceed one token at a time in the decode stage, with each step producing activations $\mathbf{X}_t \in \mathbb{R}^{B \times 1 \times C_{\text{in}}}$. At each decoding step t, POP performs fine-grained, context-conditioned pruning within the candidate region. Given the step-wise activation $\mathbf{X}_t$, channel importance scores $I_{b,i}(t)$ are evaluated independently for each sample b, only for channels $i \in \mathcal{C}$, while channels in $\mathcal{R}$ remain active and channels in $\mathcal{P}$ remain disabled. By preserving the retained region and avoiding global importance re-evaluation, POP reduces the online overhead of pruning while allowing context-conditioned adaptation. Based on the step-wise importance scores $I_{b,i}(t)$ evaluated for channels $i \in \mathcal{C}$, POP selects a subset $\mathcal{C}_{b,t} \subseteq \mathcal{C}$ for each sample such that $|\mathcal{R}| + |\mathcal{C}_{b,t}| = \text{round}((1-r)C_{\text{int}})$. This step enables fine-grained, context-conditioned pruning, as channel selection is adaptively determined by the decoding context. The selected candidate channels are combined with the retained region to form each sample's mask for decoding step t, applied before the final projection. The complete inference procedure is summarized in Algorithm 1. A PyTorch-style schematic is provided in Appendix A.6.

3.4 Pruning Metric and Inference Efficiency

In POP, we adopt the squared activation-aware importance metric introduced in Probe Pruning [41], which extends the Wanda formulation [70] to the structured pruning setting. Rather than proposing new importance formulations, we focus on enabling their efficient online use during inference.

Given intermediate activations $\mathbf{X}^{\text{int}}$ and the final FFN projection $\mathbf{W}^{\text{final}} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{int}}}$, we compute the importance of the i-th intermediate channel as

$$I_i(\mathbf{X}^{\text{int}}) = \|\mathbf{W}^{\text{final}}_{:,i}\|_4^2 \cdot \|\mathbf{X}^{\text{int}}_{:,:,i}\|_2^2. \tag{3}$$

During prefilling, this metric is applied to full-sequence intermediate FFN activations to obtain the coarse partition. During decoding, candidate-channel importance is evaluated on the step-wise intermediate activations $\mathbf{H}_t$ using the corresponding outgoing projection weights, ensuring that the scored dimension aligns with the candidate FFN channels.

To balance estimation accuracy and online efficiency, POP performs decoding in two steps. Retained and candidate channels first participate in the forward computation, yielding intermediate activations $\mathbf{H}_t = f(\mathbf{X}_t; \mathcal{R} \cup \mathcal{C})$. Step-wise importance scores $I_{b,i}(t)$ are then evaluated from $\mathbf{H}_t$ for channels $i \in \mathcal{C}$. Based on $I_{b,i}(t)$, each sample selects a subset of candidates in a context-dependent manner. Only retained channels and selected candidates are involved in the final computation for the current decoding step. Although $\mathcal{R} \cup \mathcal{C}$ participates in the first FFN layer for importance evaluation (Figure 3

Table 2: Average results for Llama-2 [72], Llama-3.1 [19], and Qwen-3 [80] across different pruning ratios on 7 QA tasks and 5 Generation tasks. The dash symbol (-) denotes non-applicable settings.

Configuration			Average QA tasks Accuracy (%) ↑								Average Generation tasks Accuracy (%) ↑							
			Llama2			Llama3.1		Qwen3			Llama2			Llama3.1		Qwen3		
PR	**Method**	**Online**	7B	13B	70B	8B	70B	8B	32B	**Avg.**	7B	13B	70B	8B	70B	8B	32B	**Avg.**
0%	N/A	N/A	59.70	63.00	66.96	64.68	69.44	65.74	68.28	65.40	28.69	33.07	45.49	44.58	61.14	60.61	55.09	46.95
20%	Wanda-sp	×	55.02	53.25	32.61	34.01	32.44	44.82	51.80	43.42	12.20	10.96	0.84	0.79	0.05	8.97	14.91	6.96
	FLAP	×	54.24	57.87	35.34	53.02	38.55	54.44	53.50	49.56	17.23	19.04	2.99	16.48	1.68	14.91	17.97	12.90
	PP	✓	57.51	60.36	66.89	54.35	67.37	-	-	61.29	19.91	27.76	41.83	19.66	51.51	-	-	32.13
	Týr	×	58.11	62.38	65.58	59.85	68.19	63.81	68.16	63.72	20.75	23.92	38.80	21.14	46.81	22.69	35.68	29.97
	POP (Ours)	✓	56.78	60.43	65.93	59.62	67.90	61.50	64.24	62.34	24.12	28.97	42.04	37.37	55.67	56.13	53.04	42.48
40%	Wanda-sp	×	41.39	37.05	32.77	33.65	31.91	34.55	40.74	36.01	4.93	2.36	0.33	0.06	0.03	0.61	5.07	1.91
	FLAP	×	45.65	47.59	33.00	43.17	34.41	36.66	40.75	40.17	9.08	14.19	0.19	6.84	0.64	0.40	6.08	5.35
	PP	✓	49.39	54.14	63.23	43.09	60.08	-	-	53.99	8.68	14.36	33.06	1.53	28.47	-	-	17.22
	Týr	×	52.01	57.88	63.73	52.01	65.90	53.73	67.21	58.92	11.27	17.09	24.92	10.31	30.24	11.22	29.60	19.24
	POP (Ours)	✓	43.95	48.15	62.58	49.26	62.53	50.98	55.98	53.35	13.84	17.99	34.47	23.34	43.14	42.73	47.78	31.90

(c)), unselected candidates are discarded before the second FFN layer and do not affect the output. The selected-channel mask follows the target ratio r, with the scoring overhead quantified in Table 8.

4 Experiment

Models and Benchmarks. We evaluate across dense LLMs (Llama-2 7B/13B/70B [72], Llama-3.1 8B/70B [19], Qwen-3 8B/32B [80]), MoE models (Qwen1.5-MoE-A2.7B [71], Qwen2-57B-A14B [79], Qwen3-30B-A3B [80]), and VLMs (Qwen2-VL-7B [74], Qwen2.5-VL-7B [6]). All evaluations use `LM-Evaluation-Harness` [23] for LLMs/MoEs and `LMMs-Eval` [86] for VLMs. Benchmarks span three categories: (i) **QA** (BoolQ [12], RTE [73], HellaSwag [84], WinoGrande [68], ARC-E/C [13], OBQA [61]), (ii) **Generation** (CoQA [67], MBPP [5], NQ-Open [40], HumanEval [10], GSM8K [14]), and (iii) **VQA** (POPE [46], OK-VQA [57], GQA [32], ScienceQA [54], MME [22]). QA and VQA tasks primarily evaluate model understanding and reasoning capability under limited-output inference settings, while generation tasks require multi-step autoregressive generation and thus provide a more challenging testbed for decode-stage efficiency. Details on metrics, full results, and perplexity evaluations are in Appendices A.4, B.12, and B.6.

Baselines and Implementation. We compare against offline structural pruning methods Wanda's [70] structural pruning variant [3], denoted as Wanda-sp, FLAP [3], Týr [44], and online Probe Pruning [41], all implemented following original designs (details in Appendix A.2). We additionally evaluate SliceGPT [4] and μ-MoE [38], with implementation details in Appendix A.3. POP uses γ=0.1 for the partition fraction. All experiments run on RTX A6000 clusters with batch sizes of 10 (LLMs/MoEs) and 1 (VLMs) in the zero-shot setting. Since POP prunes only FFN layers, we adjust FFN pruning ratios to match target parameter reduction (Appendix A.1). We additionally compare FFN-only pruning at matched structured FLOPs in Appendix B.13. Best results are shaded as first, second, and third.

4.1 Large Language Models

QA Tasks. Table 2 reports QA performance against baselines. At 20% pruning, POP achieves the second-best average performance and ranks among the top two for most evaluated models. On average, it outperforms Wanda-sp, FLAP, and Probe Pruning by 18.92%, 12.78%, and 1.05%, respectively, while trailing Týr by only 1.38%. This gap is expected, as Týr relies on offline evolutionary search with calibration data, whereas POP makes pruning decisions fully online.

Generation Tasks. Table 2 also reports generation performance. Static-mask baselines degrade severely during generation, especially on long-form tasks, where Wanda-sp and FLAP nearly collapse even at moderate pruning ratios. In contrast, POP achieves the best performance across all evaluated models and pruning ratios. At 20% pruning, it improves over Wanda-sp, FLAP, Probe Pruning, and Týr by 35.52%, 29.58%, 10.35%, and 12.51% on average, respectively. These gains highlight the importance of online pruning during autoregressive decoding. Beyond standard generation, POP remains effective in extended long-form settings, including Qwen3-Thinking mode and AIME [34]; length-binned analysis confirms its robustness across output lengths (Appendix B.9 and B.10). We additionally evaluate retrieval at context lengths up to 100K tokens (Appendix B.5). At matched FFN-channel granularity, POP improves over per-prompt μ-MoE-struct by 5.9–10.4 points; SliceGPT and weight-level comparisons are detailed in Appendix A.3.

Table 3: Performance comparison on MoE and VLM tasks. Average results (%) are shown for Qwen-MoE (1.5/2/3) and Qwen-VL (2/2.5) models across various tasks and pruning ratios (PR). The dash symbol (-) denotes non-applicable settings.

Configuration		MoE: Avg. QA Accuracy (%) ↑			MoE: Avg. Gen. Accuracy (%) ↑			VLM: VQA Accuracy (%) ↑	
PR	**Method**	Q1.5-A2.7B	Q2-A14B	Q3-A3B	Q1.5-A2.7B	Q2-A14B	Q3-A3B	Q2VL-7B	Q2.5VL-7B
0%	N/A	60.21	64.71	66.76	43.17	54.92	57.77	73.24	71.85
20%	Wanda-sp	53.97	40.61	52.37	13.21	2.35	15.71	7.96	3.99
	FLAP	53.58	65.04	32.34	20.12	37.53	0.00	52.88	47.62
	Týr	59.61	–	63.80	27.66	–	43.07	–	–
	POP (Ours)	58.40	64.64	64.28	42.10	53.95	57.96	70.86	70.48
40%	Wanda-sp	38.04	34.43	39.01	2.71	0.39	1.54	0.03	0.00
	FLAP	49.81	58.12	32.34	15.15	17.26	0.00	0.01	0.00
	Týr	48.74	–	53.74	7.81	–	15.87	–	–
	POP (Ours)	53.95	64.24	54.94	31.11	53.86	38.65	64.51	64.24

Table 4: FFN-only pruning comparison on L2-13B, L3.1-8B, and Q3-8B.

Configuration		Avg. QA Accuracy (%)				Avg. Gen. Accuracy (%)			
PR	Method	L2	L3.1	Q3	Avg	L2	L3.1	Q3	Avg
0%	Dense	63.00	64.68	65.74	64.47	33.07	44.58	60.61	46.09
20%	Wanda-sp	57.96	57.01	58.26	57.74	19.74	18.46	15.76	17.99
	FLAP	57.83	53.63	56.13	55.86	18.99	16.64	16.71	17.45
	PP	59.97	59.51	–	59.74	28.04	33.61	–	30.83
	Týr	61.10	57.61	62.99	60.57	22.53	21.56	25.80	23.30
	POP (Ours)	60.43	59.62	61.50	60.52	28.97	37.37	56.13	40.82
40%	Wanda-sp	44.30	45.77	45.04	45.04	12.92	13.34	12.58	12.95
	FLAP	49.25	44.92	36.63	43.60	13.67	12.17	4.83	10.22
	PP	51.15	50.69	–	50.92	17.28	19.95	–	18.62
	Týr	51.44	49.71	50.95	50.70	15.60	15.37	14.72	15.23
	POP (Ours)	48.15	49.26	50.98	49.46	17.99	23.34	42.73	28.02

Table 5: Attention+FFN vs. FFN-only pruning on L2-13B, L3.1-8B, and Q3-8B.

Configuration		Avg. QA Accuracy (%)				Avg. Gen. Accuracy (%)			
PR	Method	L2	L3.1	Q3	Avg	L2	L3.1	Q3	Avg
0%	Dense	63.00	64.68	65.74	64.47	33.07	44.58	60.61	46.09
20%	Wanda-sp	53.25	34.01	44.82	44.03	10.96	0.79	8.97	6.91
	FLAP	57.87	53.02	54.44	55.11	19.04	16.48	14.91	16.81
	PP	60.36	54.35	–	57.36	27.76	19.66	–	23.71
	Týr	62.38	59.85	63.81	62.01	23.92	21.14	22.69	22.58
	Ours (A+F)	61.17	56.14	61.24	59.52	29.88	23.79	50.55	34.74
40%	Wanda-sp	37.05	33.65	34.55	35.08	2.36	0.06	0.61	1.01
	FLAP	47.59	43.17	36.66	42.47	14.19	6.84	0.40	7.14
	PP	54.14	43.09	–	48.62	14.36	1.53	–	7.95
	Týr	57.88	52.01	53.73	54.54	17.09	10.31	11.22	12.87
	Ours (A+F)	53.74	41.35	46.93	47.34	17.03	4.31	12.26	11.20

4.2 Mixture-of-Expert Models and Vision Language Models

We evaluate POP on MoE models using the same protocol as dense LLMs. As shown in Table 3, POP consistently outperforms baselines on generation tasks while preserving competitive QA accuracy. At 20% pruning, POP achieves an average generation accuracy of 42.10%–57.96% across the three MoE models, substantially outperforming the best baseline Týr by up to 14.89%, while maintaining QA accuracy above 58%. At 40% pruning, baselines degrade sharply, FLAP drops to 0.00% generation accuracy on Q3-A3B, whereas POP retains 31.11%–53.86%. These results suggest that online pruning is especially beneficial for MoEs, likely due to the variability introduced by dynamic expert routing between prefill and decode stages.

We further evaluate the generalizability of POP on Qwen2-VL-7B and Qwen2.5-VL-7B across five VQA benchmarks. As shown in Table 3, baselines suffer severe degradation under pruning, often approaching zero accuracy at higher sparsity levels, while POP consistently preserves strong VQA performance. At 20% pruning, POP improves over Wanda-sp and FLAP by 62.90 and 17.98 percentage points on Qwen2-VL, and by 66.49 and 22.86 percentage points on Qwen2.5-VL, respectively. These results show that POP generalizes effectively to multimodal architectures and remains robust across application domains. Image–text calibration improves Wanda-sp but has a mixed effect on FLAP; POP remains higher on all five benchmarks at 20% PR (Appendix B.1).

4.3 Batched Inference

POP shares the prefill partition and reduced $\mathcal{R} \cup \mathcal{C}$ weights across the batch, while selecting $\mathcal{C}_{b,t}$ independently for each sample. Unselected activations are zeroed before the down projection, so both FFN projections use shared dense GEMMs; for $B > 1$, structural acceleration comes from removing $\mathcal{P}$. Table 6 compares this policy with shared candidate masks over five generation tasks. At $B = 10$, per-sample masking improves scores by 1.7 and 0.6 points on Qwen3-8B and Llama-3.1-8B, respectively.

Table 6: Mean generation score (%) over five tasks under shared and per-sample candidate masks at 20% PR.

Model	Mask policy	$B = 1$	$B = 2$	$B = 4$	$B = 10$
Qwen3-8B	Shared across batch	56.6	56.1	55.6	54.4
	Per-sample (POP)	56.6*	56.3	56.1	56.1
Llama-3.1-8B	Shared across batch	38.4	38.1	37.2	36.8
	Per-sample (POP)	38.4*	38.0	37.6	37.4

* At $B = 1$, both policies use the same input-conditioned mask.

4.4 Inference Efficiency

We profile attention and FFN latency using CUDA events and report end-to-end inference latency in Table 7. Since POP does not prune attention weights, its attention latency remains unchanged. To match the target parameter reduction, POP applies a higher pruning ratio to FFN layers, yielding larger FFN latency reductions that compensate for the absence of attention pruning. As a result, POP achieves end-to-end latency comparable to Probe Pruning, with $1.14\times$ and $1.38\times$ speedups over the dense model at 20% and 40% pruning, respectively. Appendix B.2 complements the FFN GEMM timings in Table 7 with a full latency breakdown. We further verify that POP's per-step pruning overhead remains constant across the evaluated output lengths (Appendix B.7).

Table 7: Comparison of inference latency between POP and Probe Pruning on Llama2-7B. Lat. denotes latency, and Spd. denotes speedup measured relative to the dense model.

PR	Method	Attn. (ms)		FFN (ms)		E2E (s)	
		Lat.	Spd.	Lat.	Spd.	Lat.	Spd.
0%	Dense	42	-	53	-	3.04	-
20%	PP	40	1.05×	44	1.20×	2.69	1.13×
	Ours	42	1.00×	41	1.29×	2.66	1.14×
40%	PP	37	1.14×	30	1.77×	2.14	1.42×
	Ours	42	1.00×	27	1.96×	2.21	1.38×

5 Ablation Study

FFN-only comparison and extension to attention pruning. To isolate the benefit of online pruning from the design choice of preserving attention, we first evaluate all methods in an FFN-only setting. As shown in Table 4, POP achieves the best average generation accuracy (40.82% at 20% PR, 28.02% at 40% PR), significantly outperforming Týr (23.30% and 15.23%) while remaining competitive on QA (60.52% vs. 60.57% at 20% PR). We further extend POP to jointly prune attention and FFN layers (Table 5, "Ours (A+F)"). This variant maintains solid QA performance (59.52% at 20% PR) and generation accuracy (34.74%), surpassing Týr (22.58%) and PP (23.71%). However, compared to FFN-only POP, generation quality degrades (34.74% vs. 40.82% at 20%), indicating that attention layers are more sensitive to pruning, which we leave for future work.

Matched FFN FLOPs. At matched FFN projection FLOPs of $0.80\times$ and $0.60\times$ the dense budget, POP achieves the highest mean generation scores on Llama-3.1-8B and Qwen3-8B, exceeding the strongest baseline by 10.3–28.7 points while remaining within 2.7 points of the best QA baseline. This comparison uses a contracted down projection; the FLOPs accounting and full results are provided in Appendix B.13.

Accuracy–Efficiency Trade-off under Different Pruning Strategies. We justify POP by comparing it with two variants using the same pruning metric and FFN-level structure. As shown in Table 8, Variant (1) reuses the prefill-stage mask throughout decoding. This introduces no decoding overhead but cannot adapt to evolving decoding contexts, causing accuracy degradation.

Variant (2) recomputes fine-grained pruning decisions over all channels at every decoding step. Although it improves accuracy, it incurs substantial overhead, adding over 30% FFN FLOPs during decoding and largely offsetting the efficiency gains of structural pruning.

In contrast, POP restricts online pruning to a small candidate subset, avoiding full-channel re-evaluation. With $\gamma = 0.1$, it incurs around 5% additional FFN cost while consistently improving accuracy over Variant (1), demonstrating an effective accuracy–efficiency trade-off for context-conditioned online pruning.

A correction rate analysis confirms that POP recovers approximately 28% of statically mis-pruned channels while maintaining stable partition boundaries throughout decoding (Appendix B.8).

Table 8: Accuracy-efficiency ablation under different pruning strategies. Mean accuracy is averaged over 5 tasks. FFN FLOPs are absolute for Dense and relative overhead for pruning variants.

Model	Method	Acc.	FLOPs	Overhead
L2-7B	FFN	–	22.09	–
	Var. (1)	23.40	0.00	0.0
	Var. (2)	26.56	7.32	33.1
	POP	**24.12**	**0.96**	**4.34**
L3.1-8B	FFN	–	28.75	–
	Var. (1)	34.69	0.00	0.0
	Var. (2)	38.95	9.56	33.3
	POP	**37.37**	**1.44**	**5.01**

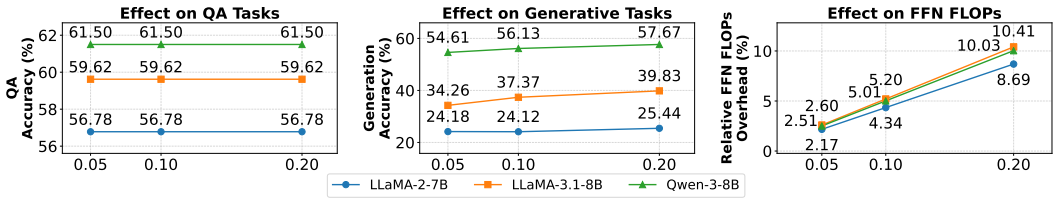

Figure 4: Ablation study on the partition fraction γ in POP. Left: mean zero-shot QA accuracy. Middle: mean generative-task accuracy. Right: relative FFN FLOPs overhead. Larger γ generally improves generation accuracy but increases overhead; we therefore set $\gamma = 0.1$ by default.

Partition Fraction. To examine candidate-region size, we vary γ and report QA accuracy, generation accuracy, and relative FFN overhead in Figure 4. Increasing γ expands the candidate region but barely affects QA, since loglikelihood-based QA has no autoregressive decoding and reduces POP to static prefilling-stage selection. Generation benefits more from larger candidate regions, which dynamically select more boundary channels under changing decoding contexts, with the clearest gains at $\gamma = 0.2$. FFN FLOPs overhead rises from 2.17–2.60% at $\gamma = 0.05$ to 8.69–10.41% at $\gamma = 0.20$. We therefore set $\gamma = 0.1$ as the default in POP to balance generation quality and efficiency (40% PR results in Appendix B.4).

Offline Preparation Cost. Table 9 compares the offline preparation cost of different pruning methods. Wanda-sp [70] and FLAP [3] require only minimal calibration, whereas Probe Pruning [41] and Týr [44] incur substantial offline preprocessing; Týr has the highest cost, exceeding one day on large-scale models such as Llama3.1-70B with four default iterations. In contrast, POP is fully online and requires no offline preparation.

Table 9: Offline preparation time (hours) comparison. Zeros indicate POP requires no preparation.

Model	W-sp	FLAP	PP	Týr	**POP**
L2-7B	0.08	0.07	1.03	9.72	**0.0**
L3.1-70B	0.43	0.38	1.42	171.93	**0.0**
Q1.5-MoE	1.14	0.85	–	45.44	**0.0**

6 Conclusion

We propose POP, an efficient online structural pruning framework that enables context-conditioned pruning with low, tunable overhead. Extensive experiments across LLMs, MoEs, and VLMs show that POP achieves competitive QA accuracy and strong generation performance against structural pruning baselines. Beyond empirical gains, POP highlights the importance of adapting pruning decisions to evolving generation contexts during autoregressive decoding, and we hope it inspires further exploration of online pruning for efficient inference of large foundation models.

Limitations. On the positive side, POP may reduce inference cost and improve access to large models in resource-constrained deployment settings. POP's QA performance can fall below offline baselines such as Týr, as online pruning does not uniformly benefit all task types. The runtime overhead of importance evaluation may also offset latency gains at low pruning ratios, and generation quality can still degrade at aggressive rates (e.g., 40% PR). POP primarily targets coupled FFN projections; standalone projections are outside its scope, and joint attention pruning is less robust. Long-context experiments cover retrieval rather than long-horizon generation, with multi-key failures at 100K tokens. The partition design is supported empirically without a formal guarantee of optimal channel selection. More broadly, structural pruning may disproportionately affect certain capabilities, potentially amplifying biases in the original model.

Acknowledgments and Disclosure of Funding

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201847, Information Technology Research Center; IITP-2026-RS-2023-00256472, Graduate School of Artificial Intelligence Semiconductor).

References

[1] Yash Akhauri, Ahmed F. AbouElhamayed, Jordan Dotzel, Zhiru Zhang, Alexander M. Rush, Safeen Huda, and Mohamed S. Abdelfattah. Shadowllm: Predictor-based contextual sparsity for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19154–19167, 2024.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35: 23716–23736, 2022.

[3] 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.

[4] 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.

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[6] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[7] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models, 2023. URL https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

[8] Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[11] 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.

[12] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 1 (Long and Short Papers), pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

[13] 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:1803.05457v1*, 2018.

[14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz 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.

[15] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[16] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

[17] Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9376–9396. Association for Computational Linguistics, 2023.

[18] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pages 5547–5569. PMLR, 2022.

[19] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[20] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[21] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

[22] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2025. URL https://arxiv.org/abs/2306.13394.

[23] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

[24] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495. Association for Computational Linguistics, 2021.

[25] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[26] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[27] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[28] Jiujun He and Huazhen Lin. Olica: Efficient structured pruning of large language models without retraining. In *International Conference on Machine Learning*. PMLR, 2025.

[29] Bairu Hou, Qibin Chen, Jianyu Wang, Guoli Yin, Chong Wang, Nan Du, Ruoming Pang, Shiyu Chang, and Tao Lei. Instruction-following pruning for large language models. In *International Conference on Machine Learning*. PMLR, 2025.

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

[31] Hanyu Hu, Pengxiang Zhao, Ping Li, Yi Zheng, Zhefeng Wang, and Xiaoming Yuan. Fasp: Fast and accurate structured pruning of large language models. *arXiv preprint arXiv:2501.09412*, 2025.

[32] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019. URL https://arxiv.org/abs/1902.09506.

[33] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[34] Maxwell Jia. Aime problem set 2024, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

[35] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[36] Lei Ke, Mingqiao Ye, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, Fisher Yu, et al. Segment anything in high quality. *Advances in Neural Information Processing Systems*, 36: 29914–29934, 2023.

[37] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[38] Toshiaki Koike-Akino, Jing Liu, and Ye Wang. mu-moe: Test-time pruning as micro-grained mixture-of-experts. *arXiv preprint arXiv:2505.18451*, 2025.

[39] Eldar Kurtić, Elias Frantar, and Dan Alistarh. Ziplm: Inference-aware structured pruning of language models. *Advances in Neural Information Processing Systems*, 36:65597–65617, 2023.

[40] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 08 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

[41] Qi Le, Enmao Diao, Ziyan Wang, Xinran Wang, Jie Ding, Li Yang, and Ali Anwar. Probe pruning: Accelerating llms through dynamic pruning via model-probing. In *International Conference on Learning Representations*, 2025.

[42] Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. Cats: Contextually-aware thresholding for sparsity in large language models. *Conference on Language Modeling*, 2024.

[43] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *International Conference on Learning Representations*, 2021.

[44] Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. Týr-the-pruner: Structural pruning llms via global sparsity distribution optimization. In *NeurIPS 2025*, 2025. URL https://neurips.cc/virtual/2025/poster/115807.

[45] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, pages 280–296. Springer, 2022.

[46] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

[47] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. *arXiv preprint arXiv:2210.06313*, 2022.

[48] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[49] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[50] Huanrong Liu, Chunlin Tian, Xuyang Wei, Jiaheng Dai, Qin Liu, Tianqi Wei, Qingbiao Li, and Li Li. Rap: Runtime-adaptive pruning for llm inference. *arXiv preprint arXiv:2505.17138*, 2025.

[51] James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *International Conference on Learning Representations*, 2025.

[52] Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics*, pages 4814–4823, 2021.

[53] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

[54] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[55] Xin Luo, Xueming Fu, Zihang Jiang, and S. Kevin Zhou. Icp: Immediate compensation pruning for mid-to-high sparsity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9487–9496, June 2025.

[56] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36:21702–21720, 2023.

[57] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge, 2019. URL https://arxiv.org/abs/1906.00067.

[58] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. *Association for Computational Linguistics*, 2(3):4, 2024.

[59] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[60] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32, 2019.

[61] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[62] Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. In *International Conference on Learning Representations*, 2024.

[63] Dev Patel, Gabrielle Gervacio, Diekola Raimi, Kevin Zhu, Ryan Lagasse, Gabriel Grand, Ashwinee Panda, and Maheep Chaudhary. Alignment-constrained dynamic pruning for llms: Identifying and preserving alignment-critical circuits. *arXiv preprint arXiv:2511.07482*, 2025.

[64] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[66] 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.

[67] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge, 2019. URL https://arxiv.org/abs/1808.07042.

[68] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

[69] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606, 2024.

[70] 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.

[71] Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

[72] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[73] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[74] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[75] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162. Association for Computational Linguistics, 2020.

[76] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. In *Association for Computational Linguistics*, 2022.

[77] 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.

[78] 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.

[79] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

[80] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[81] Mingzhe Yang, Sihao Lin, Changlin Li, and Xiaojun Chang. Let llm tell what to prune and how much to prune. In *International Conference on Machine Learning*. PMLR, 2025.

[82] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*. PMLR, 2025.

[83] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. doi: 10.1162/tacl_a_00166. URL https://aclanthology.org/Q14-1006/.

[84] 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*, 2019.

[85] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[86] Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024. URL https://arxiv.org/abs/2407.12772.

[87] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics*, pages 3013–3026. Association for Computational Linguistics, 2024.

[88] 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. *arXiv preprint arXiv:2405.18218*, 2024.

[89] Anhao Zhao, Fanghua Ye, Yingqi Fan, Junlong Tong, Zhiwei Fei, Hui Su, and Xiaoyu Shen. Skipgpt: Dynamic layer pruning reinvented with token awareness and module decoupling. *arXiv preprint arXiv:2506.04179*, 2025.

[90] 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, 2025.

[91] Yang Zhou, Zhuoming Chen, Zhaozhuo Xu, Xi V Lin, and Beidi Chen. Sirius: Contexual sparisty with correction for efficient llms. *Advances in Neural Information Processing Systems*, 37:24046–24080, 2024.

[92] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

A Implementation Details

In this section, we provide a detailed description of the experimental methodologies employed for both POP and the baseline approaches. We elaborate on the specific configurations and implementation settings to ensure a rigorous and fair evaluation.

A.1 Pruning Ratio Configurations

For the baseline methods Wanda-sp [70] and Probe Pruning [41], a uniform pruning ratio is applied to both the self-attention and MLP layers, identical to the overall target sparsity. In contrast, FLAP [3], which aggregates scores across all components, and Týr [44], which utilizes evolutionary search, dynamically allocate pruning ratios between attention and MLP layers while satisfying the global target pruning ratio.

To ensure robust performance in generative tasks, POP exclusively conducts MLP weight pruning and preserves attention mechanisms. We adjust the pruning ratio within the MLP blocks to match the total parameter reduction of these baselines for a fair comparison. Consequently, the effective sparsity applied to MLPs is inherently higher than the overall target pruning ratio. The specific effective MLP pruning ratios corresponding to each target model-level parameter reduction are detailed in Table 10. All experiments were conducted on NVIDIA A6000 48GB clusters.

Table 10: Proper MLP pruning ratios required to achieve the target model-level parameter reduction. Since our method preserves attention layers, we apply higher sparsity to MLP blocks compared to the target pruning ratio. All values are rounded to the first decimal place.

Target PR	MLP Pruning Ratio (%)											
	LLM							MoE			VLM	
	Llama2			Llama3.1		Qwen3		Qwen1.5	Qwen2	Qwen3	Qwen2	Qwen2.5
	7B	13B	70B	8B	70B	8B	32B	A2.7B	57B-A14B	30B-A3B	VL	VL
20%	29.9	29.9	24.3	24.8	24.3	25.6	24.8	29.7	22.7	30.0	22.9	22.9
40%	59.8	59.8	48.6	49.5	48.6	51.1	49.6	59.4	45.3	60.0	45.8	45.8

A.2 Offline Calibration Stage

To ensure a fair comparison, we utilized the C4 dataset [66] with 2,000 samples for Wanda-sp [70], FLAP [3], and Probe Pruning [41]. However, we distinguished the sequence length settings based on methodological requirements. For Wanda-sp and FLAP, we set the sequence length to 1,024 tokens. In contrast, for Probe Pruning, we employed a sequence length of 4,096 tokens. This setting matches the maximum position embedding of Llama2 and was necessary because Probe Pruning requires a historical state longer than the input sequence length to support generative tasks.

For Týr [44], we adhered to the configuration specified in the original literature. We used the FineWeb dataset [64] for calibration, conducting the offline stage with 1,000 samples and a maximum input length of 4k tokens, and four iterations of evolutionary search. Finally, POP operates exclusively via online pruning, eliminating the need for a separate offline calibration stage.

The original VLM comparison uses text-only C4 calibration. We additionally evaluate image–text calibration for Wanda-sp and FLAP on Qwen2.5-VL-7B to account for visual-token activation statistics (Appendix B.1).

A.3 Additional Baselines

SliceGPT [4] uses its official implementation and removes residual dimensions in a rotated basis. For μ-MoE [38], we implement per-prompt, per-row top-$(1-r)$ selection using Wanda scores, since an official implementation was not available to us during evaluation. Both are evaluated on Llama-3.1-8B and Qwen3-8B under the model-level parameter-reduction protocol of the main comparison.

To distinguish selection frequency from granularity, μ-MoE-struct applies per-prompt selection to FFN channels, whereas POP-W applies POP's per-step candidate refinement at weight granularity. Table 11 reports the five-task generation means. Weight-level results measure quality under fine-grained masking, not dense-GEMM acceleration; POP's shared structural removal reduces operator

Table 11: Generation scores (%) at 20% / 40% model-level PR. L3.1 and Q3 denote Llama-3.1-8B and Qwen3-8B. SliceGPT slices the residual dimension in a rotated basis; channel-level μ-MoE-struct and POP prune native FFN channels.

Method	Granularity	Update	L3.1	Q3
Dense	–	–	44.6	60.6
SliceGPT	Residual channel	Offline	6.9 / 1.7	18.6 / 9.2
μ-MoE	Weight	Per-prompt	41.6 / 35.4	58.9 / 56.1
POP-W	Weight	Per-step	41.2 / 36.5	58.8 / 58.0
μ-MoE-struct	FFN channel	Per-prompt	31.5 / 15.3	49.6 / 32.3
POP	FFN channel	Per-step	37.4 / 23.3	56.1 / 42.7

dimensions. SliceGPT's rotated residual channels are also different from the native FFN channels pruned by POP. These distinctions are necessary when interpreting their relative scores.

A.4 Evaluation Metrics

Table 12 presents the evaluation metrics for the tasks used in the experiments. For LLM QA tasks, we consistently used Accuracy as the primary metric. In the case of LLM Generation tasks, we employed F1 Score for CoQA, Pass@1 for MBPP and HumanEval, and Exact Match for NQ Open and GSM8K. For VLM tasks, we utilized F1 Score for POPE, Exact Match for OK-VQA, GQA, and ScienceQA, and Cognition Score for MME. To calculate the mean value for each category, we standardized all metrics to a 100-point scale. Since Accuracy, F1 Score, Pass@1, and Exact Match have a maximum value of 1, they were converted to percentages. For MME, we report the cognition score, whose maximum is 800, and convert it to a 100-point scale as $100 \times \mathrm{MME}_{\mathrm{cog}}/800$. Finally, we computed the macro average of these normalized scores to derive the mean value for each category.

Table 12: Evaluation metrics for each benchmark.

Category	**Benchmark**	**Metric**
QA	BoolQ, RTE, HellaSwag, WinoGrande, ARC-E/C, OBQA	Accuracy
Generation	CoQA	F1
	MBPP, HumanEval, NQ-Open, GSM8K	Pass@1 / Exact Match
VQA	POPE	F1
	OK-VQA, GQA, ScienceQA	Exact Match
	MME	Cognition Score
Perplexity	WikiText	Perplexity

A.5 Discussion on Activation Sparsity Methods

The design of POP builds on properties of LLM computation that are well documented in prior work on activation sparsity: a small subset of neurons is consistently important across inputs, while the remaining computation is more input- or context-dependent [62, 53, 69, 51]. We do not claim novelty for these observations. Rather, POP translates them into an online structural pruning framework.

Specifically, unlike activation-sparsity methods that suppress token-dependent activations or apply runtime thresholds [42, 91, 51], POP prunes at the structural FFN-channel granularity. This distinction leads to different deployment trade-offs. Activation-sparsity methods such as TEAL [51] can achieve strong wall-clock speedups when fine-grained dynamic sparsity is effectively supported by specialized sparse kernels or thresholding policies. By contrast, POP reduces explicit structural FLOPs by removing channels from the computation graph, while requiring no offline calibration, learned predictor, or activation-specific sparse execution.

Thus, the two directions are complementary rather than directly comparable. POP derives its tri-state partition online from each input's prefilling activations and bounds decoding-time refinement to the candidate region $\mathcal{C}$, keeping the additional FFN cost small. Future systems could combine

POP's channel-level structural reduction with activation-sparsity techniques applied to the remaining computation.

A.6 PyTorch Implementation Details

The PyTorch-style schematic below expands `Partition`, `Forward`, `Select`, `Mask`, and `Project` in Algorithm 1 for a bias-free SiLU-gated FFN. Here, `fc1` consists of the up/gate projections and activation, while `fc2` is the down projection, whose weights correspond to $\mathbf{W}^{\text{final}}$ in Eq. 3. Prefill activations can be reused from the model's forward pass.

Algorithm 1: POP: Online Structural Pruning

```
import torch, torch.nn.functional as F

# fc1 = gate/up; fc2 = W_down (= W_final in Eq.3)
# r = pruning ratio; gamma = partition fraction; k = round((1-r)·C_int)

def pop_prefill(X, W_up, W_gate, W_down, r, gamma):  # Partition(): once per prompt/batch
    X_int = F.silu(X @ W_gate.T) * (X @ W_up.T)      # Eq.(3) inputs
    I = W_down.norm(p=4, dim=0)**2 * X_int.pow(2).sum((0, 1))  # Eq.(3)
    lo = I.quantile(max(0.0, r - gamma))
    hi = I.quantile(min(1.0, r + gamma))
    R = I > hi                                    # retained, batch-shared
    C = (I >= lo) & (I <= hi)                      # candidate, batch-shared
    return R, C                                   # remaining channels are pruned

def pop_decode_step(x, W_up, W_gate, W_down, R, C):  # x: [B, 1, C_in]
    H   = fc1(x) over shared channels R ∪ C    # Forward(): dense GEMM
    s   = ||W_down[:, i]||_4^2 · H[b, i]^2, i ∈ C  # Select(): [B, |C|]
    C_t = top-(k - |R|) candidates by s[b], per sample
    H   = Mask(H, keep=R ∪ C_t, per_sample=True)  # zero unselected candidates
    return fc2(H) over shared channels R ∪ C  # use R ∪ C_t when B=1
```

The $\mathcal{R}/\mathcal{C}/\mathcal{P}$ partition is shared across the batch, while $\mathcal{C}_t$ is selected per sample. `Mask` keeps all retained activations and each sample's selected candidates while zeroing its unselected candidates before `fc2`. Batched inference uses shared $\mathcal{R} \cup \mathcal{C}$ shapes, with structured acceleration from removing $\mathcal{P}$, while $\mathcal{C}_t$ provides per-sample adaptation. For $B = 1$, `fc2` can be further contracted to $\mathcal{R} \cup \mathcal{C}_t$.

B Additional Experimental Results

B.1 Multimodal Calibration

We calibrate Wanda-sp and FLAP on Qwen2.5-VL-7B with 2,000 Flickr30k image–caption pairs [83], matching the number of calibration examples used in the C4 setting. Each pair is processed by the full VLM so that the calibration activations include visual tokens; other settings are unchanged. Table 13 reports per-benchmark scores and macro averages under both calibration settings. At 20% PR, image–text calibration improves Wanda-sp's mean from 3.99 to 48.28, but lowers FLAP's mean from 47.62 to 36.41; FLAP improves only on GQA. POP achieves 70.48, exceeding the best recalibrated baseline by 22.20 points and remaining higher on all five benchmarks. At 40% PR, both recalibrated baselines have near-zero mean scores, whereas POP retains 64.24. These results show that calibration mismatch explains part of the original baseline degradation, but does not account for the full advantage of prompt-conditioned selection.

B.2 Complete FFN Latency

We profile Llama-2-7B on an RTX A6000 with $B = 1$, generating the same 2,560 tokens in all configurations. Model-level PRs of 20% and 40% correspond to FFN channel-mask ratios of 29.9% and 59.8%. The timed loop includes candidate-band GEMMs, importance computation, top-k selection, mask/index updates, and projection weight access. Latency uses the fixed-shape implementation described in Appendix A.6; unselected candidate activations are zeroed within the shared $\mathcal{R} \cup \mathcal{C}$ dimensions.

Table 14 separates the component increments. Importance computation costs 0.007 ms, whereas top-k selection costs 0.041 ms, illustrating why arithmetic operation counts alone do not characterize online overhead. The packed weights require no additional per-step gathering; normal weight access

Table 13: Detailed VQA results for Qwen2-VL-7B and Qwen2.5-VL-7B under C4 and image–text calibration. Image–text calibration uses 2,000 Flickr30k pairs. Scores are percentages except MME cognition (MME-C, maximum 800). Mean is the macro average after mapping MME-C to a 100-point scale (score/8). POP requires no offline calibration.

Method	Calibration	PR(%)	POPE	OK-VQA	GQA	SciQA	MME-C	Mean
Qwen2-VL-7B								
Dense	–	–	88.38	52.21	62.43	85.48	621.79	73.24
Wanda-sp	C4	20	7.54	1.22	1.89	28.46	5.71	7.96
FLAP	C4	20	86.79	32.37	51.71	50.53	343.93	52.88
POP (Ours)	None	20	88.67	45.68	60.73	82.76	611.79	70.86
Wanda-sp	C4	40	0.00	0.01	0.04	0.00	0.71	0.03
FLAP	C4	40	0.00	0.00	0.01	0.05	0.00	0.01
POP (Ours)	None	40	87.99	32.06	58.13	73.14	569.64	64.51
Qwen2.5-VL-7B								
Dense	–	–	87.59	43.88	60.86	88.75	625.36	71.85
Wanda-sp	C4	20	1.93	5.75	6.80	3.51	15.71	3.99
Wanda-sp	Image–text	20	64.46	28.41	34.15	66.26	385.00	48.28
FLAP	C4	20	68.42	37.27	37.16	59.63	285.00	47.62
FLAP	Image–text	20	65.75	21.53	38.42	46.24	80.71	36.41
POP (Ours)	None	20	87.22	42.65	60.21	85.15	617.50	70.48
Wanda-sp	C4	40	0.00	0.00	0.00	0.00	0.00	0.00
Wanda-sp	Image–text	40	0.00	0.00	0.00	0.00	0.00	0.00
FLAP	C4	40	0.00	0.00	0.00	0.00	0.00	0.00
FLAP	Image–text	40	0.00	0.00	0.00	0.02	0.00	0.00
POP (Ours)	None	40	86.73	34.86	57.47	74.89	537.86	64.24

is already included in GEMM timing. The complete FFN step takes 0.379/0.265 ms, compared with 0.414 ms for dense execution. At fixed γ, candidate width remains constant with sequence length, although end-to-end runtime also depends on attention, the KV cache, and execution scheduling.

Table 14: FFN latency breakdown on Llama-2-7B, in ms per layer per decoding step ($B = 1$). Component increments are measured by adding operators to the timed computation.

Component	20% PR	40% PR
Retained-region GEMMs	0.287	0.172
Candidate-band GEMMs and masking	+0.040	+0.040
Importance computation	+0.007	+0.007
Top-k selection	+0.041	+0.041
Mask update and indexing	+0.004	+0.004
Additional per-step weight gathering	+0.000	+0.000
Complete FFN step	0.379	0.265
Dense FFN reference	0.414	0.414
Reduction vs. dense	8.5%	36.0%

B.3 Channel-Ranking Stability

We analyze Qwen3-8B and Llama-3.1-8B on GSM8K, MBPP, and CoQA across all layers. For each prompt and layer, the prefill ranking is compared with rankings from the first 128 decoding steps, covering approximately 0.5M layer–step pairs. Let T_q^{pre} and $T_{q,t}^{\mathrm{dec}}$ be the respective top-q channel sets. We define overlap as

$$O_q(t) = \frac{|T_q^{\mathrm{pre}} \cap T_{q,t}^{\mathrm{dec}}|}{|T_q^{\mathrm{pre}}|}.$$

Independent random rankings have expected overlap q, giving baselines of 0.50 and 0.10 for Top-50% and Top-10%; a shuffled control confirms these expectations. Spearman correlation compares the full rankings and has expectation zero under independence.

Table 15 shows similar task-wise patterns in both models. Top-10% overlap is 2.4–2.8 times its random expectation, while Top-50% overlap and Spearman correlation indicate considerable variation outside the most important channels. Depth also matters (Table 16): Qwen3-8B's Top-50% overlap decreases from 0.81 in early layers to 0.55 in middle layers. This motivates estimating partitions separately for each layer. The observations support retaining a high-importance core and allowing candidate refinement, without implying that all ranking changes are confined to the candidate region.

Table 15: Prefill-to-decode ranking stability averaged across layers. Entries list GSM8K / MBPP / CoQA. Random expectations are 0 for Spearman correlation, 0.50 for Top-50% overlap, and 0.10 for Top-10% overlap.

Model	Spearman	Top-50%	Top-10%
Qwen3-8B	0.23 / 0.23 / 0.22	0.58 / 0.58 / 0.58	0.26 / 0.25 / 0.25
Llama-3.1-8B	0.20 / 0.16 / 0.20	0.57 / 0.55 / 0.57	0.28 / 0.24 / 0.27

Table 16: Ranking stability averaged across tasks, grouped by depth. Entries list the first / middle / last four layers.

Model	Spearman	Top-50%	Top-10%
Qwen3-8B	0.70 / 0.15 / 0.24	0.81 / 0.55 / 0.58	0.43 / 0.22 / 0.29
Llama-3.1-8B	0.17 / 0.19 / 0.32	0.55 / 0.57 / 0.60	0.24 / 0.23 / 0.46

B.4 Partition-Fraction Robustness

Table 17: Mean generation score (%) as a function of partition fraction γ at 20% and 40% PR.

	20% PR			40% PR		
Model	.05	.10	.20	.05	.10	.20
Llama-2-7B	24.18	24.12	25.44	12.04	13.84	13.60
Llama-3.1-8B	34.26	37.37	39.83	23.05	23.34	25.88
Qwen3-8B	54.61	56.13	57.67	39.65	42.70	43.03

Table 17 evaluates $\gamma \in \{0.05, 0.10, 0.20\}$ on Llama-2-7B, Llama-3.1-8B, and Qwen3-8B at 20% and 40% PR. Increasing the candidate band generally preserves or improves generation accuracy, but does not guarantee monotonic gains: Llama-2-7B decreases slightly between $\gamma = 0.1$ and 0.2 at 40% PR. A larger candidate region also increases the width of the shared projections and the number of scores evaluated per step. The same default $\gamma = 0.1$ is used across architectures, with no model-specific tuning in the main experiments. Detailed per-task results for the original 20% sweep are in Tables 35 and 36.

B.5 Long-Context Retrieval

We evaluate Llama-3.1-8B with its native 128K context window and Qwen3-8B with YaRN scaling (factor 4, approximately 131K context). C4 documents form haystacks of 8K, 32K, 64K, and 100K tokens. Needles are inserted at relative depths 0.1, 0.5, and 0.9, with eight samples per depth and 24 samples per setting. Dense and pruned models receive identical prompts. Single-needle (SN) retrieval contains one target; multi-key (MK) retrieval includes four key–value pairs, of which three are distractors.

At 20% PR, POP matches dense Llama-3.1-8B on SN retrieval at every length (Table 18). Qwen3-8B also retains high SN accuracy, although individual settings differ from dense. MK retrieval exposes a clearer limitation: at 100K, POP's Llama-3.1-8B score is 75.0% versus 95.8% for dense, with

Table 18: Long-context retrieval accuracy (%) at 20% PR. SN denotes single-needle retrieval; MK denotes four-needle multi-key retrieval. Each condition has 24 samples.

		SN				MK			
Model	Method	8K	32K	64K	100K	8K	32K	64K	100K
Llama-3.1-8B	Dense	100.0	100.0	100.0	100.0	100.0	100.0	100.0	95.8
	POP	100.0	100.0	100.0	100.0	100.0	91.7	87.5	75.0
Qwen3-8B	Dense	100.0	100.0	100.0	83.3	100.0	100.0	95.8	91.7
(YaRN)	POP	100.0	100.0	91.7	91.7	95.8	91.7	83.3	91.7

Table 19: Wikitext [59] perplexity results for LLM and MoE models across different pruning ratios. Lower perplexity (↓) indicates better performance. Experiments are conducted using a batch size 10.

			Wikitext Perplexity (LLMs) ↓					Wikitext Perplexity (MoEs) ↓	
PR	Method	Online	Llama2			Qwen3		Qwen1.5-MoE	Qwen2
			7B	13B	70B	8B	32B	A2.7B	57B-A14B
0%	N/A	N/A	5.12	4.57	3.12	9.00	7.02	6.79	5.56
20%	Wanda-sp	×	9.23	7.90	466.51	785.16	11.55	9.92	58.76
	FLAP	×	7.53	6.64	123.74	13.99	8.23	9.64	6.02
	PP	✓	6.17	5.25	3.61	-	-	-	-
	Týr	×	6.44	5.41	4.25	11.66	7.20	7.73	-
	Ours	✓	6.86	5.94	3.92	10.95	8.44	7.51	5.75
40%	Wanda-sp	×	23.26	50.55	629.09	868.30	33.72	75.00	822.18
	FLAP	×	20.06	11.48	505.42	157.43	24.37	16.53	58.76
	PP	✓	9.82	7.34	4.62	-	-	-	-
	Týr	×	12.55	10.38	5.64	27.47	9.22	44.72	-
	Ours	✓	20.57	17.97	5.63	18.40	12.46	10.43	6.21

most observed errors involving distractor key–value confusions. On Qwen3-8B, the largest MK gap is three of 24 examples, and both models score 91.7% at 100K. These small-sample retrieval evaluations characterize long-input behavior, rather than establish performance on sustained long-horizon generation.

B.6 Perplexity

Table 19 presents perplexity scores on Wikitext [59], providing a complementary token-level view of compressed-model robustness by measuring uncertainty in next-token prediction under ground-truth contexts. A lower perplexity indicates that the pruned model maintains a distribution closer to the dense model. We find that the perplexity gap of POP is mainly concentrated on smaller models at high pruning ratios, such as Llama2-7B/13B at 40% pruning, while POP remains competitive or better on larger models and MoE architectures. This gap does not contradict POP's stronger generation accuracy, since perplexity does not perform autoregressive decoding and therefore does not fully exercise online context-conditioned pruning. Moreover, perplexity is averaged over all tokens, whereas generation accuracy reflects complete outputs and critical decoding decisions. Thus, slightly worse perplexity can still coincide with stronger downstream generation performance.

B.7 Decoding Overhead Scaling Analysis

Since POP prunes FFN channels independently at each decoding step, the importance evaluation is strictly confined to the pre-defined Candidate Region. The overhead depends only on the fixed size of the Candidate Region (determined by γ) and the current single decoding token activation $\mathbf{X}_t \in \mathbb{R}^{B\times 1\times C_{\text{in}}}$. Both factors are independent of the sequence length L, so POP's online pruning overhead remains constant per decoding step.

To verify this, we measure POP's inference latency on Qwen3-8B under 20% and 40% pruning ratios with output lengths ranging from 128 to 2048. As shown in Table 20, the per-step FFN latency remains essentially constant across all output lengths (e.g., 6.59–6.60 ms at 40% pruning), confirming that the pruning overhead does not scale with context length. The E2E speedup at 40% pruning

Table 20: Decoding latency and speedup on Qwen3-8B across different output lengths. FFN latency (ms) is measured per decoding step; E2E latency (s) is the total inference time. POP's per-step overhead remains constant regardless of sequence length.

		128		256		512		1024		2048	
	Method	Lat.	Spd.	Lat.	Spd.	Lat.	Spd.	Lat.	Spd.	Lat.	Spd.
FFN	Dense	8.33	–	8.30	–	8.40	–	8.37	–	8.40	–
	POP (20%)	7.48	1.11×	7.48	1.11×	7.48	1.12×	7.48	1.12×	7.52	1.12×
	POP (40%)	6.59	1.26×	6.56	1.27×	6.60	1.27×	6.60	1.27×	6.60	1.27×
E2E	Dense	3.16	–	6.22	–	12.87	–	24.95	–	51.41	–
	POP (20%)	3.03	1.04×	6.03	1.03×	12.48	1.03×	24.20	1.03×	50.13	1.03×
	POP (40%)	3.00	1.05×	5.95	1.05×	12.04	1.07×	23.22	1.07×	48.39	1.06×

(1.05–1.07×) is naturally smaller than the FFN-only speedup (1.26–1.27×), since attention and other non-FFN components take a larger share of total latency at longer sequences. Nevertheless, the absolute FFN latency reduction is fully preserved.

B.8 Correction Rate Analysis

Table 21: Correction Rate (%) and New Error Rate (%) on Llama-3.1-8B across decoding windows. Both metrics remain stable over decoding length.

	PR = 20%				PR = 40%			
	GSM8K		MBPP		GSM8K		MBPP	
Steps	Corr.	Err	Corr.	Err	Corr.	Err	Corr.	Err
0–49	28.40	19.41	28.33	19.46	21.78	34.46	21.78	34.57
50–99	28.38	19.43	28.07	19.79	21.75	34.47	21.45	35.34
100–149	28.24	19.61	28.00	19.89	21.56	34.86	21.35	35.58
Avg.	**28.35**	**19.47**	**28.16**	**19.69**	**21.71**	**34.57**	**21.55**	**35.10**

To quantify the effectiveness of POP's online pruning decisions, we conduct a correction rate analysis on Llama-3.1-8B (14,336 FFN channels per layer, $\gamma = 0.1$) over 150 decoding steps on GSM8K and MBPP. We report two metrics: the *Correction Rate*, measuring the fraction of statically mis-pruned channels recovered by online selection, and the *New Error Rate*, measuring channels newly excluded by online decisions that were retained under static pruning.

As shown in Table 21, POP recovers approximately 28% of statically mis-pruned channels at 20% pruning ratio and 22% at 40%. Although online selection introduces some new errors, it improves mean generation accuracy from 34.69% (static Variant 1 in Table 8) to 37.37%, suggesting that recovered channels contribute more than the new errors detract. Full-channel re-evaluation (Variant 2) achieves 38.95%.

Both metrics remain stable across three decoding windows (std $< 0.25\%$ for Correction Rate and $<$ 0.6% for New Error Rate), showing stable recovery statistics within the evaluated 150-step horizon.

B.9 Long-Form Generation and Output-Length Analysis

We further evaluate POP on MBPP, HumanEval, GSM8K, and AIME using both Qwen3-8B/32B (non-thinking) and Qwen3-8B/32B-Thinking (thinking mode, which produces significantly longer outputs). All main paper results for Qwen3 use the non-thinking mode.

As shown in Table 22, POP consistently outperforms all baselines across both model sizes and modes at 20% pruning ratio. On generation tasks, POP achieves 65.75% and 47.18% on 8B and 8B-T, far exceeding Týr's 11.66% and 14.61%. On 32B models, POP achieves generation scores of 60.43% and 65.10%, substantially outperforming Týr (31.98% and 55.25%). For AIME, POP is the only method achieving non-zero accuracy on 8B-T (3.33%) and obtains 66.67% on 32B-T, surpassing Týr's 63.33%.

Table 22: Long-form generation and AIME results at 20% pruning ratio on Qwen3 models. "T" denotes Thinking mode. POP consistently outperforms all baselines across both standard and thinking modes.

		Avg. Generation (%)				AIME (%)			
PR	**Method**	8B	8B-T	32B	32B-T	8B	8B-T	32B	32B-T
0%	N/A	71.26	67.22	63.49	71.59	23.33	73.33	46.67	73.33
20%	Wanda-sp	0.53	0.40	3.35	2.32	0.00	0.00	0.00	0.00
	FLAP	0.63	0.73	1.74	19.54	0.00	0.00	0.00	0.00
	Týr	11.66	14.61	31.98	55.25	0.00	0.00	26.67	63.33
	Ours	**65.75**	**47.18**	**60.43**	**65.10**	**0.00**	**3.33**	**30.00**	**66.67**

B.10 Length Binning Analysis

Table 23: Length-binned accuracy on Qwen3-32B-Thinking at 20% pruning ratio. POP maintains strong performance across all output-length bins.

# Tokens (AIME)	Dense	Týr	POP
0–5000	1.000	1.000	1.000
5000–10000	1.000	0.800	0.800
10000–15000	1.000	1.000	1.000
>15000	0.333	0.308	**0.400**

# Tokens (MBPP)	Dense	Týr	POP
0–1500	0.790	0.504	**0.864**
1500–3000	0.710	0.714	**0.789**
>3000	**0.330**	0.215	0.241

We additionally analyze accuracy by output length using binned evaluation on Qwen3-32B-Thinking. As shown in Table 23, for AIME, POP achieves the highest accuracy in the longest bin (>15000 tokens: 0.400 vs. Týr's 0.308). For MBPP, POP substantially outperforms Týr in short-to-medium bins (0–1500: 0.864 vs. 0.504) and still outperforms Týr in longer outputs (0.241 vs. 0.215). These results confirm that POP's advantage is robust across varying output lengths.

B.11 Segment Anything Model

The Segment Anything Model (SAM) [37] comprises a heavy image encoder, a prompt encoder, and a lightweight mask decoder. Notably, the image encoder accounts for over 90% of the parameters and dominates inference costs, making it a prime candidate for pruning-based acceleration. To validate the extensibility of POP to this pure vision architecture, we adapted our method to target the image encoder. Unlike autoregressive models that utilize a token decoding loop, SAM operates via a single-pass representation; consequently, we applied POP exclusively using the strategy designed for the *prefill stage*. Given the lack of established structured pruning baselines for this architecture, we implemented a *Random structured pruning* baseline for comparison.

We evaluated zero-shot segmentation performance on the full validation sets to ensure statistical significance: 5,000 images for COCO [48] and 19,809 for LVIS [25]. All experiments were conducted with a batch size of 10. To generate box prompts, we adopted the generators employed in HQ-SAM [36]: FocalNet-DINO [85] for COCO and ViTDet-H [45] for LVIS. The detailed experimental results are presented in Tables 24 and 25. These results demonstrate that POP effectively maintains performance compared to dense and random baselines, confirming its efficacy in pure vision architectures.

B.12 Detailed Benchmark Results

In this section, we present detailed experimental results stratified by task. Our evaluation encompasses dense models, POP models, and various baseline methods.

Tables 26, 27, 28, and 32 report the performance on zero-shot commonsense question-answering (QA) benchmarks for Large Language Models (LLMs) and Mixture-of-Experts (MoE) models. For text generation tasks, the results across LLMs and MoE models are provided in Tables 29, 30, 31, and 33. We also evaluate the language modeling capability of the models.

Moreover, Table 13 illustrates the experimental outcomes for Visual Question Answering (VQA) tasks using Vision-Language Models (VLMs).

Table 24: Detailed Accuracies (%) on COCO dataset [48] using SAM models with different levels of structured sparsity. We compare the dense model with POP and Random structured pruning at 20% sparsity. For the COCO dataset, we use a SOTA detector FocalNet-DINO [85] trained on the COCO dataset as our box prompt generator. Bold indicates the better performance between POP and Random.

Model	Method	PR (%)	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP_{avg}
ViT-B	Dense	–	46.16	73.17	49.21	34.61	50.52	59.32	52.16
	Random	20	44.31	71.76	46.36	32.02	48.66	57.86	50.16
	POP (Ours)	20	**45.38**	**72.79**	**47.94**	**34.06**	**49.57**	**58.32**	**51.34**
ViT-L	Dense	–	48.72	74.97	52.94	36.85	53.47	61.90	54.81
	Random	20	47.18	73.66	**50.70**	35.01	**51.96**	**60.55**	53.18
	POP (Ours)	20	**47.23**	**74.14**	50.68	**35.88**	51.71	59.93	**53.26**
ViT-H	Dense	–	49.03	75.20	53.23	37.13	54.14	61.78	55.09
	Random	20	**47.70**	73.98	**51.68**	35.52	**52.82**	**61.25**	**53.82**
	POP (Ours)	20	47.56	**74.23**	50.99	**36.09**	52.08	60.32	53.54

Table 25: Detailed Accuracies (%) on LVIS dataset [25] using SAM models with different levels of structured sparsity. We compare the dense model with POP and Random structured pruning at 20% sparsity. For LVIS, we adopt ViTDet-H [45] trained on the LVIS dataset as our box prompt generator. Bold indicates the better performance between POP and Random.

Model	Method	PR (%)	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP_r	AP_c	AP_f	AP_{avg}
ViT-B	Dense	–	41.41	59.61	43.68	29.47	53.49	61.39	32.96	42.27	44.18	45.38
	Random	20	39.20	58.00	40.90	27.61	50.65	59.10	31.05	40.50	41.34	43.15
	POP (Ours)	20	**41.07**	**59.28**	**43.51**	**29.56**	**52.87**	**60.49**	**32.60**	**41.99**	**43.76**	**45.02**
ViT-L	Dense	–	44.01	60.78	46.63	31.48	56.77	65.46	33.84	44.94	47.44	47.93
	Random	20	42.26	59.72	44.56	29.70	54.79	**64.15**	32.75	43.31	45.26	46.28
	POP (Ours)	20	**43.34**	**60.58**	**46.03**	**31.83**	**55.43**	63.01	**33.87**	**44.20**	**46.54**	**47.20**
ViT-H	Dense	–	44.65	61.15	47.38	32.09	57.54	65.68	34.61	45.43	48.19	48.53
	Random	20	42.82	60.06	45.37	30.23	55.26	**64.07**	34.10	43.55	45.86	46.81
	POP (Ours)	20	**43.66**	**60.68**	**46.37**	**31.76**	**55.77**	63.87	**34.14**	**44.31**	**47.13**	**47.52**

Finally, Tables 34, 35, and 36 present the detailed results corresponding to the ablation study discussed in Section 5. Table 34 compares the performance of models under different pruning strategies. Tables 35 and 36 demonstrate the impact of the partition fraction, which is a hyperparameter of POP, on model accuracy for QA tasks and generative tasks, respectively.

Table 26: Detailed Accuracies (%) of Llama-2 [72] for 7 QA tasks with different levels of structured sparsity.

Params	Method	PR(%)	BoolQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
Llama2-7B	Dense	-	77.71	62.82	57.17	69.14	76.26	43.43	31.40	59.7
	Wanda-sp	20	68.53	53.79	54.74	67.25	72.18	39.85	28.80	55.02
	FLAP	20	68.90	61.01	51.89	66.06	66.50	36.35	29.00	54.24
	Probe Pruning	20	72.51	61.01	54.75	66.61	73.99	42.07	31.60	57.51
	Týr	20	75.60	62.46	53.94	68.35	74.71	41.30	30.40	58.11
	POP (Ours)	20	72.63	63.54	53.42	67.72	71.13	39.59	29.40	56.78
	Wanda-sp	40	61.74	54.51	35.08	51.14	49.37	23.12	14.80	41.39
	FLAP	40	63.39	50.54	42.92	62.59	49.03	27.05	24.00	45.65
	Probe Pruning	40	64.43	45.85	46.68	58.72	66.25	34.98	28.80	49.39
	Týr	40	69.27	49.46	46.44	64.01	68.98	37.12	28.80	52.01
	POP (Ours)	40	64.04	50.18	37.71	57.46	50.76	27.73	19.80	43.95
Llama2-13B	Dense	-	80.55	65.34	60.05	72.14	79.42	48.29	35.20	63
	Wanda-sp	20	67.46	54.15	48.56	67.48	69.74	37.54	27.80	53.25
	FLAP	20	73.49	60.65	55.98	71.90	71.13	42.15	29.80	57.87
	Probe Pruning	20	78.78	59.21	58.45	70.32	77.32	44.63	33.80	60.36
	Týr	20	79.82	68.59	58.01	73.01	77.65	46.16	33.40	62.38
	POP (Ours)	20	79.94	67.87	57.68	69.61	74.29	43.00	30.60	60.43
	Wanda-sp	40	62.17	52.71	30.07	51.46	31.52	20.05	11.40	37.05
	FLAP	40	63.43	53.07	42.00	61.64	55.14	31.06	26.80	47.59
	Probe Pruning	40	70.40	49.82	52.77	62.51	71.25	40.44	31.80	54.14
	Týr	40	76.91	61.37	53.03	68.82	73.86	40.36	30.80	57.88
	POP (Ours)	40	68.23	56.32	42.32	60.62	55.51	32.42	21.60	48.15
Llama2-70B	Dense	-	83.76	67.87	64.78	77.98	82.70	54.44	37.20	66.96
	Wanda-sp	20	37.83	51.99	26.84	49.49	29.42	19.71	13.00	32.61
	FLAP	20	50.55	49.46	30.01	48.38	34.55	20.05	14.40	35.34
	Probe Pruning	20	82.26	69.68	64.04	78.14	82.41	55.29	36.40	66.89
	Týr	20	77.10	69.68	63.80	75.85	81.31	54.35	37.00	65.58
	POP (Ours)	20	82.94	66.07	63.92	76.48	81.69	54.18	36.20	65.93
	Wanda-sp	40	38.23	54.87	26.24	49.80	27.23	21.42	11.60	32.77
	FLAP	40	39.63	54.87	26.49	49.80	26.05	20.99	13.20	33
	Probe Pruning	40	80.58	63.90	61.42	72.77	78.70	51.62	33.60	63.23
	Týr	40	76.15	69.31	60.63	76.56	78.49	49.15	35.80	63.73
	POP (Ours)	40	82.60	64.98	60.31	73.72	77.74	46.08	32.60	62.58

Table 27: Detailed Accuracies (%) of Llama-3.1 [19] for 7 QA tasks with different levels of structured sparsity.

Params	Method	PR(%)	BoolQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
Llama3.1-8B	Dense	-	81.96	70.76	60.01	73.72	81.48	51.45	33.40	64.68
	Wanda-sp	20	46.12	54.15	26.62	49.41	29.29	19.11	13.40	34.01
	FLAP	20	68.62	61.37	48.26	68.27	65.15	32.85	26.60	53.02
	Probe Pruning	20	67.92	53.79	53.02	63.30	73.36	40.44	28.60	54.35
	Týr	20	78.20	65.34	54.31	68.43	76.18	44.71	31.80	59.85
	POP (Ours)	20	77.28	63.90	54.68	70.88	76.22	44.20	30.20	59.62
	Wanda-sp	40	45.75	50.54	25.63	49.96	28.20	20.65	14.80	33.65
	FLAP	40	62.05	57.76	37.35	58.41	44.91	20.90	20.80	43.17
	Probe Pruning	40	58.62	53.07	33.78	54.85	55.98	25.51	19.80	43.09
	Týr	40	71.35	59.93	44.27	61.80	68.01	33.11	25.60	52.01
	POP (Ours)	40	66.30	56.68	40.44	62.75	61.95	35.07	21.60	49.26
Llama3.1-70B	Dense	-	85.38	69.68	66.43	79.24	87.21	60.92	37.20	69.44
	Wanda-sp	20	38.59	51.99	25.88	51.14	25.46	20.22	13.80	32.44
	FLAP	20	57.03	53.07	29.53	53.67	36.28	19.88	20.40	38.55
	Probe Pruning	20	85.60	66.07	63.82	78.30	84.72	58.28	34.80	67.37
	Týr	20	85.72	69.31	64.67	79.16	85.10	58.36	35.00	68.19
	POP (Ours)	20	84.25	68.95	65.32	78.14	85.86	58.36	34.40	67.9
	Wanda-sp	40	37.86	53.79	26.00	47.51	24.66	21.93	11.60	31.91
	FLAP	40	44.92	52.71	27.93	49.57	31.73	18.60	15.40	34.41
	Probe Pruning	40	77.98	62.46	57.10	70.48	78.24	45.48	28.80	60.08
	Týr	40	84.07	70.76	60.64	77.03	81.48	53.75	33.60	65.9
	POP (Ours)	40	78.78	67.15	58.81	72.46	80.09	50.85	29.60	62.53

Table 28: Detailed Accuracies (%) of Qwen3 [80] for 7 QA tasks with different levels of structured sparsity.

Params	Method	PR(%)	BoolQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
Qwen3-8B	Dense	-	86.61	77.98	57.10	68.11	83.29	55.46	31.60	65.74
	Wanda-sp	20	64.59	53.07	37.64	52.57	54.46	26.62	24.80	44.82
	FLAP	20	77.52	66.07	50.05	62.90	64.39	35.32	24.80	54.44
	Týr	20	84.62	74.73	54.27	67.56	81.61	52.47	31.40	63.81
	POP (Ours)	20	84.77	74.37	51.76	64.80	77.78	48.21	28.80	61.5
	Wanda-sp	40	48.41	48.74	27.65	50.20	34.39	18.86	13.60	34.55
	FLAP	40	57.62	51.63	32.31	51.07	32.28	19.54	12.20	36.66
	Týr	40	62.39	67.15	42.98	59.35	73.70	39.51	31.00	53.73
	POP (Ours)	40	76.67	65.70	39.52	57.14	60.31	32.34	25.20	50.98
Qwen3-32B	Dense	-	86.30	76.17	63.91	73.32	84.43	57.86	36.00	68.28
	Wanda-sp	20	72.39	59.57	49.21	59.12	63.47	36.86	22.00	51.8
	FLAP	20	85.96	79.06	61.13	67.40	25.08	22.70	33.20	53.5
	Týr	20	86.97	78.34	63.94	71.90	83.38	57.00	35.60	68.16
	POP (Ours)	20	85.72	76.53	60.94	65.43	76.85	52.22	32.00	64.24
	Wanda-sp	40	59.51	51.26	36.80	51.54	42.72	25.34	18.00	40.74
	FLAP	40	58.10	53.79	39.37	51.22	42.76	23.04	17.00	40.75
	Týr	40	85.75	75.45	61.08	70.64	83.71	57.85	36.00	67.21
	POP (Ours)	40	80.67	70.40	51.23	58.33	64.52	40.10	26.60	55.98

Table 29: Accuracies (%) of Llama-2 [72] for 5 Generative tasks with different levels of structured sparsity.

Params	Method	PR	coqa	mbpp	nq_open	humaneval	gsm8k	Mean
Llama2-7B	Dense	-	77.35	23.40	18.89	9.76	14.03	28.69
	Wanda-sp	20	51.37	0.00	7.56	0.00	2.05	12.2
	FLAP	20	73.77	2.00	6.21	1.22	2.96	17.23
	Probe Pruning	20	66.46	8.40	10.80	6.71	7.20	19.91
	Týr	20	75.46	8.20	11.27	1.83	6.98	20.75
	POP (Ours)	20	76.46	16.20	9.78	7.93	10.24	24.12
	Wanda-sp	40	22.45	0.00	0.28	0.00	1.90	4.93
	FLAP	40	43.07	0.00	0.42	0.00	1.90	9.08
	Probe Pruning	40	38.18	1.40	2.38	0.00	1.44	8.68
	Týr	40	51.69	0.00	4.64	0.00	0.00	11.27
	POP (Ours)	40	56.44	3.80	2.02	3.05	3.87	13.84
Llama2-13B	Dense	-	79.18	27.20	23.63	12.20	23.12	33.07
	Wanda-sp	20	49.96	0.60	1.66	0.00	2.58	10.96
	FLAP	20	76.94	3.40	6.54	0.61	7.73	19.04
	Probe Pruning	20	76.05	20.80	15.43	10.37	16.15	27.76
	Týr	20	78.05	11.40	11.63	1.83	16.68	23.92
	POP (Ours)	20	79.00	21.60	13.85	9.76	20.62	28.97
	Wanda-sp	40	10.65	0.00	0.00	0.00	1.14	2.36
	FLAP	40	68.49	0.00	1.11	0.00	1.36	14.19
	Probe Pruning	40	52.36	5.20	5.18	3.66	5.38	14.36
	Týr	40	74.79	2.00	5.15	0.00	3.49	17.09
	POP (Ours)	40	59.51	10.00	2.91	6.10	11.45	17.99
Llama2-70B	Dense	-	83.71	45.00	29.00	16.46	53.30	45.49
	Wanda-sp	20	2.68	0.00	0.00	0.00	1.52	0.84
	FLAP	20	14.13	0.00	0.00	0.00	0.83	2.99
	Probe Pruning	20	82.24	39.00	23.30	14.02	50.57	41.83
	Týr	20	81.66	30.20	24.32	10.98	46.85	38.8
	POP (Ours)	20	83.09	38.00	23.82	14.63	50.64	42.04
	Wanda-sp	40	0.99	0.00	0.00	0.00	0.68	0.33
	FLAP	40	0.67	0.00	0.00	0.00	0.30	0.19
	Probe Pruning	40	79.98	25.60	14.07	10.98	34.65	33.06
	Týr	40	75.85	0.00	13.24	5.49	30.02	24.92
	POP (Ours)	40	81.71	24.00	11.52	14.02	41.09	34.47

Table 30: Accuracies (%) of Llama-3.1 [19] for 5 Generative tasks with different levels of structured sparsity.

Params	Method	PR	coqa	mbpp	nq_open	humaneval	gsm8k	Mean
Llama3.1-8B	Dense	-	80.88	48.40	7.70	35.37	50.57	44.58
	Wanda-sp	20	3.12	0.00	0.00	0.00	0.83	0.79
	FLAP	20	73.01	1.80	2.83	1.22	3.56	16.48
	Probe Pruning	20	54.45	17.60	8.20	11.59	6.44	19.66
	Týr	20	75.52	8.60	4.85	7.93	8.79	21.14
	POP (Ours)	20	79.70	39.20	3.82	23.78	40.33	37.37
	Wanda-sp	40	0.00	0.00	0.06	0.00	0.23	0.06
	FLAP	40	32.16	0.00	0.00	0.00	2.05	6.84
	Probe Pruning	40	5.38	0.00	0.30	0.61	1.36	1.53
	Týr	40	45.51	0.00	4.15	0.00	1.90	10.31
	POP (Ours)	40	69.20	14.40	2.63	12.20	18.27	23.34
Llama3.1-70B	Dense	-	83.70	64.60	24.96	51.22	81.20	61.14
	Wanda-sp	20	0.19	0.00	0.06	0.00	0.00	0.05
	FLAP	20	7.04	0.00	0.06	0.00	1.29	1.68
	Probe Pruning	20	81.27	54.60	20.19	26.83	74.68	51.51
	Týr	20	82.51	47.20	20.64	15.85	67.85	46.81
	POP (Ours)	20	83.11	59.20	17.40	40.85	77.79	55.67
	Wanda-sp	40	0.07	0.00	0.00	0.00	0.08	0.03
	FLAP	40	2.58	0.00	0.03	0.00	0.61	0.64
	Probe Pruning	40	68.80	25.40	9.36	7.93	30.86	28.47
	Týr	40	78.85	15.40	8.25	5.49	43.22	30.24
	POP (Ours)	40	80.24	40.60	10.25	25.00	59.59	43.14

Table 31: Accuracies (%) of Qwen3 [80] for 5 Generative tasks with different levels of structured sparsity.

Params	Method	PR	coqa	mbpp	nq_open	humaneval	gsm8k	Mean
Qwen3-8B	Dense	-	81.97	65.60	7.31	60.98	87.19	60.61
	Wanda-sp	20	43.16	0.00	0.11	0.00	1.59	8.97
	FLAP	20	72.11	0.00	0.55	0.00	1.90	14.91
	Týr	20	75.17	3.20	3.32	0.61	31.16	22.69
	POP (Ours)	20	81.05	56.00	2.35	56.10	85.14	56.13
	Wanda-sp	40	1.71	0.00	0.00	0.00	1.36	0.61
	FLAP	40	1.64	0.00	0.14	0.00	0.23	0.4
	Týr	40	48.71	0.00	5.32	0.00	2.05	11.22
	POP (Ours)	40	73.73	31.80	0.89	35.37	71.87	42.73
Qwen3-32B	Dense	-	82.45	78.20	2.52	37.81	74.45	55.09
	Wanda-sp	20	63.39	0.20	1.11	0.61	9.25	14.91
	FLAP	20	81.01	3.40	3.63	1.83	0.00	17.97
	Týr	20	81.31	44.00	1.14	0.00	51.93	35.68
	POP (Ours)	20	80.75	70.00	3.16	39.63	71.65	53.04
	Wanda-sp	40	24.21	0.00	0.08	0.00	1.06	5.07
	FLAP	40	29.31	0.00	0.03	0.00	1.06	6.08
	Týr	40	79.55	0.00	2.69	1.22	64.52	29.6
	POP (Ours)	40	75.98	49.20	1.63	37.81	74.30	47.78

Table 32: Detailed Accuracies (%) of Qwen1.5-MoE [71], Qwen2-MoE [79] and Qwen3-MoE [80] models for 7 QA tasks with different levels of structured sparsity.

Params	Method	PR(%)	BoolQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
Qwen1.5-MoE A2.7B	Dense	-	79.73	68.59	57.98	69.53	73.06	41.81	30.80	60.21
	Wanda-sp	20	68.32	59.93	52.99	58.64	72.35	39.93	25.60	53.97
	FLAP	20	72.66	64.62	47.43	67.32	63.68	32.17	27.20	53.58
	Týr	20	76.64	69.31	53.56	69.77	74.96	41.81	31.20	59.61
	POP (Ours)	20	77.65	68.95	55.67	68.04	70.96	39.16	28.40	58.4
	Wanda-sp	40	62.17	52.71	29.56	53.20	36.79	19.28	12.60	38.04
	FLAP	40	64.04	62.09	42.89	64.17	59.85	30.46	25.20	49.81
	Týr	40	64.45	53.43	36.34	56.28	70.71	35.58	24.40	48.74
	POP (Ours)	40	74.89	62.09	47.21	66.69	66.50	35.24	25.00	53.95
Qwen2 57B-A14B	Dense	-	86.36	75.09	62.93	74.03	75.08	46.67	32.80	64.71
	Wanda-sp	20	59.17	53.07	32.63	53.51	46.97	23.55	15.40	40.61
	FLAP	20	87.25	75.45	62.78	75.14	76.60	46.08	32.00	65.04
	POP (Ours)	20	87.00	75.45	62.37	73.95	74.75	47.18	31.80	64.64
	Wanda-sp	40	48.20	52.71	27.15	49.65	27.19	21.93	14.20	34.43
	FLAP	40	73.73	67.51	55.80	68.27	70.92	41.38	29.20	58.12
	POP (Ours)	40	85.60	75.09	60.28	74.67	75.30	47.53	31.20	64.24
Qwen3 30B-A3B	Dense	-	88.75	82.31	59.56	70.56	79.29	52.82	34.00	66.76
	Wanda-sp	20	73.30	63.54	44.73	60.62	62.21	37.37	24.80	52.37
	FLAP	20	37.83	47.29	25.59	50.12	24.33	21.42	19.80	32.34
	Týr	20	86.33	77.26	53.10	68.90	77.36	48.04	35.60	63.8
	POP (Ours)	20	87.28	81.59	55.00	68.43	75.25	48.04	34.40	64.28
	Wanda-sp	40	56.70	50.18	34.49	49.49	41.71	23.72	16.80	39.01
	FLAP	40	37.83	47.29	25.59	50.12	24.33	21.42	19.80	32.34
	Týr	40	65.26	60.29	43.03	61.48	73.78	41.72	30.60	53.74
	POP (Ours)	40	79.63	74.73	41.72	63.54	63.13	36.43	25.40	54.94

Table 33: Detailed Accuracies (%) of Qwen1.5-MoE [71], Qwen2-MoE [79] and Qwen3-MoE [80] models for 5 Generative tasks with different levels of structured sparsity.

Params	Method	PR	coqa	mbpp	nq_open	humaneval	gsm8k	Mean
Qwen1.5-MoE A2.7B	Dense	-	78.61	38.80	4.57	32.93	60.96	43.17
	Wanda-sp	20	57.46	0.00	5.87	0.61	2.12	13.21
	FLAP	20	75.76	5.20	5.04	5.49	9.10	20.12
	Týr	20	73.77	14.00	3.24	7.32	39.96	27.66
	POP (Ours)	20	79.49	32.60	5.26	32.93	60.20	42.1
	Wanda-sp	40	12.53	0.00	0.03	0.00	0.99	2.71
	FLAP	40	67.51	1.40	3.66	0.61	2.58	15.15
	Týr	40	33.56	0.00	3.57	0.00	1.90	7.81
	POP (Ours)	40	76.99	21.20	2.24	12.20	42.91	31.11
Qwen2 57B-A14B	Dense	-	82.13	60.00	0.64	51.83	79.99	54.92
	Wanda-sp	20	9.87	0.00	0.30	0.00	1.59	2.35
	FLAP	20	81.76	22.60	1.25	11.59	70.43	37.53
	POP (Ours)	20	81.81	59.60	0.72	48.17	79.45	53.95
	Wanda-sp	40	0.18	0.00	0.03	0.00	1.74	0.39
	FLAP	40	66.15	1.00	1.77	0.61	16.76	17.26
	POP (Ours)	40	81.77	56.80	1.61	47.56	81.58	53.86
Qwen3 30B-A3B	Dense	-	81.70	73.00	12.08	32.93	89.16	57.77
	Wanda-sp	20	53.71	0.40	1.14	3.66	19.64	15.71
	FLAP	20	0.00	0.00	0.00	0.00	0.00	0
	Týr	20	79.75	45.60	4.96	20.73	64.29	43.07
	POP (Ours)	20	80.84	66.40	8.37	45.73	88.48	57.96
	Wanda-sp	40	6.82	0.00	0.06	0.61	0.23	1.54
	FLAP	40	0.00	0.00	0.00	0.00	0.00	0
	Týr	40	58.65	7.00	2.16	6.10	5.46	15.87
	POP (Ours)	40	79.85	28.60	1.77	15.24	67.78	38.65

Table 34: Detailed accuracies (%) of Llama-2 [72] and Llama-3.1 [19] on five generative tasks for the accuracy-efficiency ablation summarized in Table 8.

Params	Method	PR	coqa	mbpp	nq_open	humaneval	gsm8k	Mean
Llama3.1-8B	Variant (1)	20	78.06	32.4	3.60	21.95	37.45	34.69
	POP (Ours)	20	79.70	39.20	3.82	23.78	40.33	37.37
	Variant (2)	20	78.10	43.2	5.60	24.39	43.44	38.95
Llama2-7B	Variant (1)	20	75.83	15.20	7.87	6.71	11.37	23.4
	POP (Ours)	20	76.46	16.20	9.78	7.93	10.24	24.12
	Variant (2)	20	76.99	19.80	14.57	9.76	11.68	26.56

Table 35: Ablation study on the effect of partition fraction in POP on QA tasks. Accuracies (%) are reported under a fixed pruning ratio.

Params	PF	BoolQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
Llama2-7B	0.05	71.84	63.18	52.17	66.46	70.03	38.14	28.60	55.77
	0.10	72.63	63.54	53.42	67.72	71.13	39.59	29.40	56.78
	0.20	72.29	64.26	55.06	67.80	72.31	40.02	28.40	57.16
Llama3.1-8B	0.05	76.42	66.79	53.14	70.09	75.13	43.09	29.00	59.09
	0.10	77.28	63.90	54.68	70.88	76.22	44.20	30.20	59.62
	0.20	79.14	67.15	57.05	71.43	78.87	49.23	33.20	62.3
Qwen3-8B	0.05	84.34	74.01	50.13	64.33	76.31	47.10	30.20	60.92
	0.10	84.77	74.37	51.76	64.80	77.78	48.21	28.80	61.5
	0.20	85.41	74.37	54.89	65.19	78.83	52.05	30.60	63.05

Table 36: Ablation study on the effect of partition fraction in POP on generative tasks. Accuracies (%) are reported under a fixed pruning ratio.

Params	PF	CoQA	MBPP	NQ-open	HumanEval	GSM8K	Mean
Llama2-7B	0.05	77.14	15.00	9.00	8.54	11.22	24.18
	0.10	76.46	16.20	9.78	7.93	10.24	24.12
	0.20	76.65	18.20	12.44	9.15	10.77	25.44
Llama3.1-8B	0.05	78.47	29.00	35.40	3.42	25.00	34.26
	0.10	79.70	39.20	3.82	23.78	40.33	37.37
	0.20	78.63	43.40	6.59	28.05	42.46	39.83
Qwen3-8B	0.05	80.73	52.20	1.19	53.66	85.29	54.61
	0.10	81.05	56.00	2.35	56.10	85.14	56.13
	0.20	80.01	61.80	4.32	57.32	84.91	57.67

B.13 FFN-Only Comparison at Matched FLOPs

To compare methods under the same pruning scope and computation budget, we restrict all methods to FFN-only pruning and keep the remaining components unchanged. We match the structured FFN projection FLOPs at $0.80\times$ and $0.60\times$ the dense budget. For this comparison, the gate/up projections operate over $\mathcal{R} \cup \mathcal{C}$ and the down projection over $\mathcal{R} \cup \mathcal{C}_t$, giving relative FFN FLOPs

$$f(r) = \frac{2(1 - r + \gamma) + (1 - r)}{3},$$

where r is the FFN channel-mask ratio and $\gamma = 0.1$. This accounting uses a contracted down projection; the fixed-shape implementation profiled in Appendix B.2 retains the shared $\mathcal{R} \cup \mathcal{C}$ dimension for that projection.

Table 37 reports QA and generation results on Llama-3.1-8B and Qwen3-8B. POP achieves the highest mean generation score at both budgets on both models, while its QA score remains within 2.7 points of the best baseline.

Table 37: FFN-only pruning at matched structured FLOPs. Budgets are relative to dense FFN projection FLOPs under the contracted-down accounting defined above. QA and generation are mean scores (%). L3.1 and Q3 denote Llama-3.1-8B and Qwen3-8B.

		QA		Generation	
FFN budget	Method	L3.1	Q3	L3.1	Q3
$0.80\times$	Wanda-sp	57.0	58.3	18.5	15.8
	FLAP	53.6	56.1	16.6	16.7
	Týr	57.6	63.0	21.6	25.8
	POP	57.5	60.3	36.4	54.5
$0.60\times$	Wanda-sp	45.8	45.0	13.3	12.6
	FLAP	44.9	36.6	12.2	4.8
	Týr	49.7	51.0	15.4	14.7
	POP	48.6	49.3	25.7	40.7

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 clearly state the contributions: an online structural pruning framework (POP) that adapts pruning decisions per decoding step within a bounded candidate region. These claims are supported by experiments across LLMs, MoEs, and VLMs in Section 4 and ablation studies in Section 5. Limitations are discussed in Section 6.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include a dedicated Limitations paragraph in Section 6, discussing: (1) QA performance can fall below offline baselines, (2) runtime overhead may offset gains at low pruning ratios, (3) generation quality degrades at aggressive pruning rates, and (4) FFN-focused scope and long-context retrieval failures limit generality.

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 results or proofs. The method is motivated by empirical observations of contextual sparsity (Section 3.2) and validated experimentally.

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: We provide full implementation details including the pruning metric (Section 3.4), hyperparameters ($\gamma = 0.1$), hardware setup (RTX A6000), batch sizes, evaluation frameworks, and all benchmark configurations. Algorithm 1 specifies partitioning and shared-shape decoding, with a PyTorch reference in Appendix A.6. Additional details are in Appendices A.2 and A.1.

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 code is available at `https://github.com/Yi1-Chen/POP`. All models used are publicly available (Llama-2, Llama-3.1, Qwen-3, Qwen-VL series), and all benchmarks are publicly accessible through `LM-Evaluation-Harness` and `LMMs-Eval`.

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: POP is a training-free method. All inference-time settings are specified in Section 4, including partition fraction γ, pruning ratios, batch sizes, and hardware. Pruning ratio configurations and baseline calibration settings are detailed in Appendices A.1 and A.2.

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: We do not report confidence intervals or repeated-run error bars. Evaluation follows the benchmark protocols, and Appendix B.5 specifies the sample count for the retrieval experiments. Their small sample sizes limit the precision of those estimates.

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: We report that all experiments are conducted on RTX A6000 GPU clusters (Section 4). Complete FFN latency and its component breakdown are provided in Appendix B.2.

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: This work focuses on improving inference efficiency of publicly available foundation models. It does not involve human subjects, private data, or dual-use concerns beyond those inherent to the underlying models.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The Limitations paragraph in Section 6 discusses potential positive impacts through lower-cost deployment and potential negative impacts from uneven capability degradation or bias amplification.

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: POP is a pruning method applied to existing publicly available models. It does not release new pre-trained models or datasets that pose additional misuse risks beyond those of the original models.

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 models and benchmarks are properly cited throughout the paper. The models used (Llama-2, Llama-3.1, Qwen-3, Qwen-VL) and evaluation frameworks (`LM-Evaluation-Harness`, `LMMs-Eval`) are publicly released under their respective licenses.

13. **New assets**

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

    Answer: [Yes]

Justification: The paper does not release new datasets or pre-trained models. Code and documentation are available at https://github.com/Yi1-Chen/POP.

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: This work does not involve crowdsourcing or research with human subjects.

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: This work does not involve human subjects research.

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 are not used as a component of the core method. LLMs serve as evaluation targets (models being pruned), not as tools in the methodology itself.