POP: Prefill-Only Pruning for Efficient Large Model Inference

Junhui He[1,2] **Zhihui Fu**[2] **Jun Wang**[2] **Qingan Li**[1] *

[1]Wuhan University [2]OPPO Research Institute

Abstract

Large Language Models (LLMs) and Vision-Language Models (VLMs) have demonstrated remarkable capabilities. However, their deployment is hindered by significant computational costs. Existing structured pruning methods, while hardware-efficient, often suffer from significant accuracy degradation. In this paper, we argue that this failure stems from a stage-agnostic pruning approach that overlooks the asymmetric roles between the prefill and decode stages. By introducing a virtual gate mechanism, our importance analysis reveals that deep layers are critical for next-token prediction (decode) but largely redundant for context encoding (prefill). Leveraging this insight, we propose Prefill-Only Pruning (POP), a stage-aware inference strategy that safely omits deep layers during the computationally intensive prefill stage while retaining the full model for the sensitive decode stage. To enable the transition between stages, we introduce independent Key-Value (KV) projections to maintain cache integrity, and a boundary handling strategy to ensure the accuracy of the first generated token. Extensive experiments on Llama-3.1, Qwen3-VL, and Gemma-3 across diverse modalities demonstrate that POP achieves up to 1.37× speedup in prefill latency with minimal performance loss, effectively overcoming the accuracy-efficiency trade-off limitations of existing structured pruning methods.

1 Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) have achieved remarkable success across various domains. However, their massive parameter counts impose substantial computational overhead during inference, limiting their deployment. To solve this challenge, model pruning has been explored as a means to remove redundant computation and accelerate inference. While unstructured pruning methods (Frantar and Alistarh, 2023; Sun et al., 2024) can preserve accuracy, they often require specialized hardware and kernels to realize speedups. Conversely, structured pruning methods (Ma et al., 2023; Ashkboos et al., 2024; Men et al., 2025; Yang et al., 2024b; Song et al., 2024), which remove entire components like layers or channels, offer better hardware compatibility but often suffer from significant accuracy degradation, particularly in open-ended generative tasks.

We argue that the failure of existing structured pruning methods stems from a stage-agnostic, "one-size-fits-all" approach that ignores the functional asymmetry of the inference process. Standard autoregressive inference consists of two distinct stages: prefill and decode. The prefill stage aims solely to encode the input history into Key-Value (KV) cache to provide context for future generation. In contrast, the decode stage has a dual role: it must encode the current token into the cache, while simultaneously modeling the probability distribution of the next token. Intuitively, these distinct roles imply different sensitivities to pruning, requiring an asymmetric pruning strategy.

Motivated by this intuition, we propose **Prefill-Only Pruning (POP)**, a novel strategy that accelerates the computationally intensive prefill stage while preserving the full model capacity for the sensitive decode stage. We first introduce the virtual gate mechanism for layer importance estimation, by approximating the loss increment on the calibration dataset when each layer is removed. Then, we analyze the importance of layers during prefill and decode stage respectively (depicted in Section 3.2, Figure 1), and uncover a striking disparity: deep layers are critical for the generation phase but are largely redundant for the context encoding phase. Leveraging this insight, we accelerate inference by pruning these deep layers exclusively during the prefill stage, while retaining the full model capacity

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 5864–5877
July 2-7, 2026 ©2026 Association for Computational Linguistics

for the decode stage. To ensure a seamless transition between the pruned and full stages, we further incorporate mechanisms to handle the missing KV states and the stage boundary.

Our main contributions are summarized as follows:

- We introduce a virtual gate mechanism to model the importance of each layer to the final loss, revealing the functional asymmetry of LLMs: deep layers are essential for decode but redundant for prefill.
- We propose Prefill-Only Pruning (POP), a stage-aware method that removes deep layers during prefill to reduce FLOPs, while retaining the full model for decode. We employ independent KV projections to generate KV states for the pruned layers, and boundary handling to ensure the accuracy of the first generated token.
- We conduct extensive evaluations across diverse model families (Llama-3.1, Qwen3-VL, Gemma-3) and modalities. Experimental results demonstrate that POP achieves significant speedups with minimal accuracy loss, effectively overcoming the limitations of stage-agnostic structured pruning.

2 Preliminary

2.1 Transformer Inference and KV Cache

We consider a standard decoder-only Transformer architecture (Vaswani et al., 2017; Team, 2024; Bai et al., 2025; Team, 2025). Let L denote the number of layers in the model. For a specific layer $l \in \{1, \dots, L\}$, let $x_l \in \mathbb{R}^d$ denote the input hidden state (where d is the hidden dimension). The computation within layer l typically consists of a Grouped-Query Attention (GQA) block or a Multi-Head Latent Attention (MLA) Block, followed by a Feed-Forward Network (FFN) block, both with residual connections and layer normalization.

The forward pass for the l-th layer can be expressed as:

$$\begin{aligned} y_l &:= x_l + \mathrm{Attn}(x_l, K_l^{\text{past}}, V_l^{\text{past}}) \\ x_{l+1} &:= y_l + \mathrm{FFN}(y_l) \end{aligned} \tag{1}$$

where K_l^{past} and V_l^{past} represent the cached Keys and Values from previous tokens in the sequence.

KV Cache Generation. During the inference process, specifically for the attention mechanism, the model computes the Query (q_l), Key (k_l), and Value (v_l) for the current token using projection matrices W_l^Q, W_l^K, W_l^V. To capture positional information, Rotary Positional Embeddings (RoPE) are typically applied to the Queries and Keys. The computation for the new KV pairs of the current token is:

$$\begin{aligned} k_l^{\text{new}} &:= \mathrm{RoPE}(\mathrm{LN}(x_l) W_l^K) \\ v_l^{\text{new}} &:= \mathrm{LN}(x_l) W_l^V \end{aligned} \tag{2}$$

where $\mathrm{LN}(\cdot)$ denotes the normalization layer.

To enable autoregressive generation without recomputing history, these new keys and values are appended to the cache:

$$\begin{aligned} K_l^{\text{current}} &:= \mathrm{Concat}(K_l^{\text{past}}, k_l^{\text{new}}) \\ V_l^{\text{current}} &:= \mathrm{Concat}(V_l^{\text{past}}, v_l^{\text{new}}) \end{aligned} \tag{3}$$

The attention output is then computed using the updated K_l^{current} and V_l^{current}.

2.2 Layer Pruning Formulation

Layer pruning aims to accelerate inference by removing entire layers—both the Attention and FFN blocks—while preserving the residual connections. Formally, let $S_{\text{skip}} \subset \{1, \dots, L\}$ be the set of indices representing the layers to be pruned. For any layer $l \in S_{\text{skip}}$, we bypass the computational blocks entirely. The propagation through a pruned layer is reduced to an identity mapping:

$$\hat{x}_{l+1} := x_l, \quad \forall l \in S_{\text{skip}} \tag{4}$$

In existing pruning approaches, the set S_{skip} is applied in a stage-agnostic manner across both the prefill and decode stage. However, as we discuss in the following section, this approach ignores the asymmetrical functional goals of the two phases: the prefill phase focuses solely on context encoding, while the decoding phase focuses on both context encoding and next-token prediction.

3 Method

3.1 Estimating Layer Importance with Virtual Gates

To effectively identify and remove redundant computation, we first require a quantitative metric to measure the contribution of each layer to the model's overall performance. Intuitively, we define the importance score of the l-th layer as the increment of average loss on a calibration dataset when the layer is removed (pruned), while keeping other parameters unchanged. We denote this importance score as I_l.

Calculating I_l directly based on this definition by physically removing each layer and evaluating the model is computationally intensive, requiring L separate inference passes for an L-layer model. To address this, we introduce **virtual gates**. We modify the forward pass of the l-th layer by multiplying the residual branches (Attention and FFN outputs) with a virtual scalar parameter g_l (Molchanov et al., 2019):

$$\begin{aligned} \hat{y}_l &:= x_l + \mathrm{Attn}(x_l, K_l^{\text{past}}, V_l^{\text{past}}) \odot g_l \\ \hat{x}_{l+1} &:= \hat{y}_l + \mathrm{FFN}(\hat{y}_l) \odot g_l \end{aligned} \tag{5}$$

When $g_l = 1$, the layer functions identically to the original pre-trained model; when $g_l = 0$, the residual update is suppressed, and $\hat{x}_{l+1} = x_l$, effectively pruning the layer.

We estimate the importance score I_l by approximating the change in loss $\mathcal{L}$ when g_l shifts from 1 to 0, using a second-order Taylor expansion around $g_l = 1$ (LeCun et al., 1989; Molchanov et al., 2019):

$$\begin{aligned} I_l &= \mathbb{E}\left[\mathcal{L}_{g_l=0} - \mathcal{L}_{g_l=1}\right] \\ &\approx \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial g_l}(0-1) + \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial g_l^2}(0-1)^2\right] \\ &= \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial g_l}\right] + \frac{1}{2}\mathbb{E}\left[\frac{\partial^2 \mathcal{L}}{\partial g_l^2}\right] \end{aligned} \tag{6}$$

Calculating the second-order term $\frac{\partial^2 \mathcal{L}}{\partial g_l^2}$ directly is still computationally intensive. To approximate this term efficiently, we leverage the properties of Fisher Information. Specifically, we adopt a sampling-based strategy to satisfy the assumptions linking the Hessian to the gradient variance (Kunstner et al., 2019). For each prompt x in the calibration dataset, instead of using ground-truth targets, we sample the target response $\hat{y} \sim P_\theta(\cdot|x)$ from the model's distribution to compute the loss. This approach aligns the data distribution with the model distribution, achieving two key simplifications for calculating I_l:

Vanishing First-Order Term: Since the model minimizes loss on its own generated distribution, the expected first-order gradient is zero:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial g_l}\right] = 0 \tag{7}$$

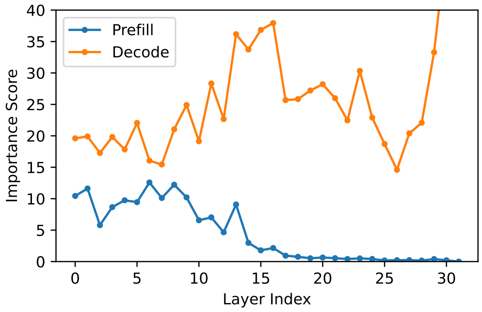
Hessian-Gradient Relation: Under this sampling strategy, the expected Hessian matches the second moment of the gradients (i.e., the Fisher Information Matrix):

$$\mathbb{E}\left[\frac{\partial^2 \mathcal{L}}{\partial g_l^2}\right] = \mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial g_l}\right)^2\right] \tag{8}$$

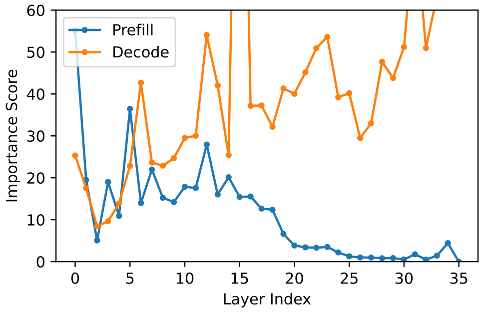
By substituting these simplifications back into the Taylor expansion, we derive an efficient estimator that relies solely on the gradient computed during a single forward-backward pass on each calibration sample:

$$\tilde{I}_l = \mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial g_l}\right)^2\right] \tag{9}$$

By estimating layer importance with virtual gates, we accurately capture the sensitivity of the model outputs to the pruning of specific layers, while avoiding the iterative removal of each layer, or the heavy computation of the second-order derivative.



(a) Llama-3.1-8B-Instruct, WizardLM-V2-196K



(b) Qwen3-VL-8B-Instruct, LLAVA-Instruct-150K

Figure 1: Importance scores of layers from different models over datasets.

3.2 Stage-Aware Importance Analysis

Consider the standard inference process of large models, which consists of two distinct stages: prefill and decode. The prefill stage has a singular role: to process the user's input prompt $x_{1:N-1}$ in parallel and encode the token information into the KV cache of every layer, providing context for future generation. In contrast, the decode stage processes the single latest token x_t at each step. It must simultaneously fulfill a dual role: (1) encode the current token into the KV cache and append it to the sequence history; (2) model the probability distribution of the next token x_{t+1} for autoregressive generation.

Despite sharing the same model parameters θ, the functional roles of these two stages are asymmetric. Intuitively, these two stages might require different pruning strategies. This motivates us to investigate the following questions:

RQ1: Do prefill and decode stages exhibit asymmetric sensitivity to pruning? Does one stage exhibit consistently higher sensitivity compared to the other, indicating greater fragility to pruning?

RQ2: Do specific layers exhibit stage-dependent redundancy? Are there any layers that play critical roles in one stage, while being redundant in the other?

To answer these questions, we estimate the importance score of each layer during prefill and decode stage respectively, by extending the virtual gate mechanism to be stage-aware.

Specifically, we treat the gates for the prefill and decode stages as separate parameters, denoted as g^{prefill} and g^{decode}, respectively. Let $\mathcal{E}$ and $\mathcal{D}$ represent the prefill and decoding processes. The prefill stage takes the input prompt $x_{1:N-1}$ and outputs the KV cache for the context:

$$Z_{1:N-1} = \{(K_l, V_l)\}_{l=1}^{L} := \mathcal{E}_{\theta, g^{\text{prefill}}}(x_{1:N-1}) \tag{10}$$

The decode stage takes the current token x_t, the generated history $x_{N:t-1}$, and the past KV cache $Z_{1:N-1}$ (from prefill) to predict the probability distribution of x_{t+1}:

$$P_\theta(x_{t+1}|x_{1:t}) := \mathcal{D}_{\theta, g^{\text{decode}}}(x_{t+1} \,|\mathcal{E}_{\theta, g^{\text{prefill}}}(x_{1:N-1}), x_{N:t}) \tag{11}$$

The final loss for the sequence is the cross-entropy over all output tokens:

$$\mathcal{L} = -\sum_{t=N}^{T-1} \log \mathcal{D}_{\theta, g^{\text{decode}}}(x_{t+1} \,|\mathcal{E}_{\theta, g^{\text{prefill}}}(x_{1:N-1}), x_{N:t}) \tag{12}$$

We calculate the gradients for g^{prefill} and g^{decode} via a single forward-backward pass on each calibration sample, to obtain the stage-specific importance scores:

$$\begin{aligned} \tilde{I}_l^{\text{prefill}} &= \mathbb{E}[(\partial \mathcal{L}/\partial g_l^{\text{prefill}})^2] \\ \tilde{I}_l^{\text{decode}} &= \mathbb{E}[(\partial \mathcal{L}/\partial g_l^{\text{decode}})^2] \end{aligned} \tag{13}$$

We conducted experiments using Llama-3.1-8B-Instruct on the text dataset WizardLM-V2-196k, and Qwen3-VL-8B-Instruct on the multimodal dataset LLAVA-Instruct-150K. The calculated importance scores are visualized in Figure 1. From experimental results, we observe consistent characteristics across different models and modalities:

Disparity between stages: The importance scores for the prefill and decode stages are highly asymmetric. For a majority of layers, the decode importance (orange line) is significantly higher than the prefill importance (blue line), indicating that the decode stage is much more sensitive to model pruning.

Criticality of Deep Layers for the Decode Stage: For decode stage, deep layers are generally more important than shallow layers. Specifically, for the orange line, the first few layers show moderate importance, while importance increases with depth. The final layers exhibit extremely high scores, often exceeding the visualization range, indicating criticality for next-token prediction.

Redundancy of Deep Layers for the Prefill Stage: The layer importance distribution of prefill stage is markedly different. For the blue line, the initial layers show moderate importance, indicating they are crucial for initial feature extraction. The intermediate layers show a decline in importance. Notably, the final layers exhibit low importance scores, often approaching zero, indicating redundancy for later generation.

These results validate our hypothesis: there is a disparity in overall sensitivity between stages; deep layers are essential for constructing the output distribution (decode) but are largely redundant for encoding the context information (prefill). These observations motivate us to propose **prefill-only pruning (POP), a stage-aware pruning method that improves prefill efficiency while preserving model accuracy**.

3.3 Prefill-Only Pruning for Efficient Inference

Based on the stage-aware importance profile in Section 3.2, we adopt a static pruning strategy removing the deep layers in prefill stage, while retaining the full model for decode stage. Specifically, we prune the last 1/3 of the layers, a ratio empirically selected to balance efficiency and accuracy. Extensive experiments on the sensitivity to pruning ratio are presented in Section 4.4.

Implementation of this asymmetric strategy requires addressing two key challenges: missing KV caches for pruned layers, and the boundary handling between the prefill and decode stage.

KV Cache Generation with Independent KV Projections. A naive skipping of layer l during prefill would result in a missing KV cache (K_l, V_l). Since the decode stage uses the full model, it requires valid KV entries for all layers to perform attention over the input history.

To resolve this, we decouple the KV projection from the main computational block. For a pruned layer $l \in S_{\text{skip}}$ during prefill, we execute independent KV projections by: (1) Compute KV Cache: We apply the projection matrices to the input state x_l to generate and store the cache:

$$k_l := \mathrm{RoPE}(\mathrm{LN}(x_l)W^K), \quad v_l := \mathrm{LN}(x_l)W^V \tag{14}$$

(2) Skip computation: We bypass the heavy Attention and FFN computations, passing the input directly to the next layer:

$$\hat{x}_{l+1} := x_l \tag{15}$$

Since the computational and memory access cost of the projections (W^K, W^V) is negligible compared to the full Attention and FFN blocks ($< 5\%$ for Llama-3, Qwen-3 and gemma-3 models), this method maintains the speedup benefits of pruning while ensuring the decoding stage has access to a complete KV cache.

A potential concern regarding independent KV projections is the representation mismatch for deep layers. We provide further experiments and discussions in Appendix A to address this concern.

Boundary Handling for the Last Input Token. In a standard inference pipeline, the prefill stage processes tokens $x_{1:N}$ and predicts x_{N+1}. However, our analysis shows that deep layers are critical for next-token prediction. If we prune the deep layers when processing the last input token x_N, the accuracy of the first generated token will be degraded, leading to an accumulation of errors throughout the entire generation process.

To mitigate this, we redefine the boundary between stages. We define the **pruned prefill stage** as processing $x_{1:N-1}$. The processing of the last input token x_N is treated as the **first decode step**. This ensures that the prediction of the first new token utilizes the model's full capacity, while improving the efficiency of the computationally intensive prefill stage.

4 Experiments

4.1 Experimental Setup

To comprehensively evaluate the effectiveness and generalization of our proposed POP, we conduct experiments across various model architectures, modalities, and downstream tasks.

Models. We select a diverse set of state-of-the-art open-weights models to demonstrate the generalization capability of our approach. Our selection covers both text-only models and vision-language models from different model series and sizes:

- **Text-Only Models:** We utilize Llama-3.1-8B-Instruct (Team, 2024) to evaluate performance on text understanding and generation tasks.
- **Vision-Language Models:** We utilize Qwen3-VL-8B-Instruct (Bai et al., 2025) and Gemma-3-12B-It (Team, 2025) to evaluate performance on text understanding, text generation and vision understanding tasks.

Methods. We compare POP with representative unstructured and structured pruning methods:

- **Unstructured**: We compare with Wanda (Sun et al., 2024), an unstructured weight pruning method based on weight magnitudes and input activations.
- **Structured**: We compare with two methods: (1) SliceGPT (Ashkboos et al., 2024): removes rows and columns of weight matrices using PCA-based transformations; and (2) ShortGPT (Men et al., 2025): identifies and removes redundant layers based on cosine similarities of hidden states.

To ensure a fair comparison, we adjust the pruning ratio of all baselines to achieve a comparable FLOPs reduction during the prefill stage.

Benchmarks. We employ a diverse set of benchmarks covering common sense reasoning, generative tasks, contextual understanding, and multimodal capabilities:

- **Common Sense:** We report 0-shot accuracy on MMLU (Hendrycks et al., 2021), HellaSwag(Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), and PIQA (Bisk et al., 2020).
- **Math & Code:** We evaluate complex reasoning capabilities using GSM8K (Cobbe et al.,

Table 1: **Accuracy comparison across different models and tasks.** "Avg" denotes the average score across all tasks. The pruning ratios are indicated in parentheses. † denotes likelihood-based tasks; ‡ denotes open-ended generation tasks. **Bold** indicates the best results for all structured pruning methods. *Italic* indicates unstructured pruning methods (Wanda).

Method	**Common Sense**†				**Math & Code**‡		**Long Context QA**‡		**Multi-Modal**‡				**Avg**
	MMLU	HellaSwag	WinoG	PIQA	GSM8K	HumanEval	MultiFieldQA	HotpotQA	MMMU	RealWorldQA	TextVQA	ScreenSpot	
Llama-3.1-8B-Instruct													
Full Model	68.33	79.50	74.40	81.12	79.68	68.29	54.57	55.66	-	-	-	-	70.19
Wanda (30%)	*65.87*	*78.96*	*74.59*	*80.74*	*76.42*	*65.84*	*52.80*	*53.03*	-	-	-	-	*68.53*
SliceGPT (25%)	34.97	51.19	66.54	63.87	0.91	0.00	12.35	8.71	-	-	-	-	29.82
ShortGPT (25%)	65.80	61.93	69.77	70.51	0.38	0.00	6.80	3.81	-	-	-	-	34.88
POP (31.25%)	**67.43**	**78.29**	**73.40**	**80.36**	**77.26**	**64.63**	**52.88**	**53.48**	-	-	-	-	**68.47**
Qwen3-VL-8B-Instruct													
Full Model	74.95	76.60	73.72	79.92	81.50	92.07	53.53	65.49	51.33	69.67	82.24	87.03	74.00
Wanda (30%)	*73.78*	*75.22*	*72.45*	*80.47*	*83.32*	*90.85*	*52.87*	*63.19*	*52.00*	*67.45*	*81.08*	*85.22*	*73.16*
SliceGPT (25%)	39.16	44.50	57.93	67.25	13.95	17.68	40.76	38.33	28.00	32.55	13.54	0.24	32.82
ShortGPT (25%)	33.85	48.24	61.56	64.96	0.83	0.00	21.44	16.37	32.22	53.07	33.69	0.86	30.59
POP (33.3%)	**75.05**	**76.44**	**73.88**	**80.14**	**80.21**	**89.63**	**52.34**	**63.13**	**50.67**	**69.28**	**80.73**	**86.40**	**73.16**
Gemma-3-12B-It													
Full Model	71.46	81.96	74.35	78.07	73.62	82.32	55.90	59.62	46.78	54.64	67.02	11.08	63.07
Wanda (30%)	*69.70*	*80.82*	*73.64*	*77.42*	*75.13*	*83.54*	*55.28*	*58.78*	*45.89*	*55.29*	*64.67*	*10.38*	*62.55*
SliceGPT (25%)	22.95	34.12	54.14	55.93	1.67	0.00	10.83	4.18	25.56	5.23	2.59	0.24	18.12
ShortGPT (25%)	23.81	30.32	48.70	53.70	0.91	0.00	1.58	0.34	25.00	0.39	0.00	0.24	15.42
POP (33.3%)	**71.37**	**81.96**	**74.59**	**79.76**	**73.16**	**81.10**	**57.33**	**59.11**	**46.78**	**55.42**	**63.71**	**11.08**	**62.95**

2021) for mathematics and HumanEval (Chen et al., 2021) for code generation.

- **Long Context QA:** We evaluate long context understanding capabilities using MultiFieldQA for single-doc QA and HotpotQA for multi-doc QA (Bai et al., 2024).
- **Multimodal Understanding:** For VLM evaluation, we use MMMU (Yue et al., 2024) for multi-discipline understanding, RealWorldQA (xAI, 2024) for spatial reasoning, TextVQA (Singh et al., 2019) for OCR-based QA, and ScreenSpot (Cheng et al., 2024) for GUI element localization.

For accuracy evaluations, we adopt two distinct strategies on different tasks. For common sense reasoning, we employ a likelihood-based approach: the model ranks candidate options based on their conditional probabilities (normalized by length), selecting the highest-scoring option as the prediction. Conversely, for other tasks, we utilize an open-ended generation approach: the model produces full responses via greedy decoding, which are then evaluated using exact match or functional correctness after rule-based answer extraction.

Implementation Details. Calibration datasets for all methods consist of 200 samples from the WizardLM-V2-196K dataset (Xu et al., 2024) for text-only models, or the LLAVA-Instruct-150K dataset (Liu et al., 2023) for vision-language models. All experiments are implemented in PyTorch (Paszke et al., 2019; Wu, 2023) using the HuggingFace Transformers (Wolf et al., 2019) library and executed on NVIDIA A100 80GB GPUs. Evaluation on downstream tasks are conducted using the LM-Evaluation-Harness (Gao et al., 2024) library, the LongBench library (Bai et al., 2024) and the LMMs-Eval library (Zhang et al., 2025b).

4.2 Accuracy on Downstream Tasks

Table 1 compares the pruning ratios and accuracies of different methods. Experimental results draw the following conclusions:

Existing structured pruning methods exhibit catastrophic collapse on open-ended generation tasks. As shown in Table 1, while SliceGPT and ShortGPT maintain reasonable performance on likelihood-based tasks, they suffer from severe accuracy degradation on open-ended generation tasks. For instance, when applied to Llama-3.1, SliceGPT drops from 79.68% to 0.91% on GSM8K. Similarly, on the multimodal Qwen3-VL, SliceGPT degrades ScreenSpot accuracy from 87.03% to 0.86%. These results suggest that existing structured pruning methods destroy the generation capability of models.

POP preserves model accuracies across benchmarks. In contrast, POP demonstrates remarkable stability across all task categories, despite pruning a larger portion of the model ($\approx 33\%$) compared to the baselines ($\approx 25\%$). More specifically, for generative reasoning tasks, POP achieves 77.26% on GSM8K and 64.63% on HumanEval when applied to Llama-3.1, retaining 97.00% and 95.64% of the full model's performance, respectively. For long context QA tasks, POP also exhibits minimal performance drops (e.g., 59.11% vs 59.62%

on HotpotQA when applied to Gemma-3). The robustness extends to multimodal models and tasks. On Qwen3-VL, POP maintains near-lossless performance on MMMU (50.67% vs 51.33%) and ScreenSpot (86.40% vs 87.03%), significantly outperforming structured pruning baselines.

POP achieves accuracies comparable to unstructured pruning methods, while offering better hardware compatibility. Wanda, being an unstructured pruning method, generally preserves accuracy better than traditional structured methods. However, unstructured pruning methods requires specialized hardware and kernels for acceleration. POP achieves accuracy on par with Wanda across benchmarks (e.g., Gemma-3 Avg: 62.95% vs 62.55%) while offering much better hardware compatibility by structurally removing model layers.

4.3 Inference Speedup

We evaluate the inference speedup of POP by measuring the Time-to-First-Token (TTFT) on NVIDIA A100 GPUs. We conduct all experiments with a batch size of 8, utilizing text inputs with lengths ranging from 32 to 2048 tokens, and image inputs with resolutions ranging from 640×480 to 2560×1440. Experimental results are shown in Table 2.

Hardware Limitations for Unstructured Pruning. While Wanda achieves high accuracy on downstream tasks, it yields no wall-clock speedup ($1.0\times$) on our GPUs (A100) using dense kernels. This result confirms that unstructured pruning theoretically reduces FLOPs but requires specialized hardware and sparse kernels to realize efficiency gains.

Impact of Sequence Length for Text Inputs. For text inputs, we observe that the efficiency gains of POP are highly dependent on the input sequence length. At short context lengths (e.g., 32 tokens), POP exhibits limited speedups (e.g., $1.22\times$ for Llama-3.1, $1.02\times$ for Gemma-3). This is primarily due to our boundary handling strategy. The short-input prefill is a memory-bound process, dominated by model weight access. Since processing the final input token requires using the full model, POP cannot reduce these memory access overheads, thus limiting performance gains.

Table 2: **TTFT speedup comparison across different models and input lengths.** All experiments are conducted with a batch size of 8. Values represent the speedup ratio relative to the full model ($1.0\times$).

Llama-3.1-8B-Instruct

Method \ Input Length	32	128	512	2048
Wanda	1.00	1.00	1.00	1.00
SliceGPT	1.22	**1.31**	1.29	1.31
ShortGPT	**1.30**	1.29	1.31	1.30
POP	1.22	1.27	**1.34**	**1.36**

Gemma-3-12B-It

Method \ Input Length	32	128	512	2048
Wanda	1.00	1.00	1.00	1.00
SliceGPT	1.10	**1.29**	1.27	1.29
ShortGPT	**1.25**	**1.29**	1.31	1.31
POP	1.02	1.27	**1.34**	**1.37**

Qwen3-VL-8B-Instruct

Method \ Resolution	640×480	1280×720	1920×1080	2560×1440
Wanda	1.00	1.00	1.00	1.00
SliceGPT	1.14	1.16	1.15	1.14
ShortGPT	1.18	1.17	1.15	1.13
POP	**1.19**	**1.19**	**1.18**	**1.16**

However, as the sequence length increases, the computational cost of the first $N-1$ tokens (processed by the pruned model) becomes the dominant factor in TTFT. Consequently, POP demonstrates significant speedup. At an input length of 2048, POP achieves a $1.36\times$ speedup on Llama-3.1 and $1.37\times$ on Gemma-3, outperforming both SliceGPT and ShortGPT. These results confirm that POP is particularly well-suited for compute-bound, long-context scenarios.

Efficiency on Multimodal Tasks. For vision inputs, POP delivers speedups between $1.16\times$ and $1.19\times$, consistently surpassing SliceGPT and ShortGPT for all image resolutions, while offering much better accuracies. These results confirm the advantage of POP in multimodal tasks.

Overall, experimental results validate that POP offers a practical "plug-and-play" acceleration solution that requires no model retraining or specialized hardware or kernels, making it particularly advantageous for long-context and high-resolution multimodal processing where prefill latency is critical.

4.4 Ablation Study

To validate the design choices and parameter sensitivity of POP, we conduct comprehensive ablation studies using Qwen3-VL. We report the accuracy on GSM8K (complex reasoning) and HotpotQA (long-context understanding).

4.4.1 Effectiveness of Design Choices

We first verify the necessity of our three key design components: (1) targeting deep layers, (2) independent KV projections, and (3) boundary handling. Experimental results are shown in Table 3.

Layer Selection Strategy. We compare POP against the shallow pruning (first 1/3 layers) and interleaved pruning (every 3$^{\text{rd}}$ layer) strategies. Both variants suffer from significant accuracy degrada-

Table 3: **Ablation on design choices.** We compare POP with different layer selection strategies and component removals on Qwen3-VL-8B-Instruct. “w/o Indep. KV” denotes removing independent KV projections for pruned layers. “w/o Boundary” denotes removing the boundary handling for the last input token.

Method Variants	**GSM8K**	**HotpotQA**
Full Model	81.50	65.49
POP	**80.21**	**63.13**
Layer Selection Strategy		
Shallow Pruning	0.15	0.00
Interleaved Pruning	56.48	6.81
Component Necessity		
w/o Indep. KV Proj.	2.05	1.18
w/o Boundary Handling	77.33	11.45

tion, with shallow pruning dropping to nearly zero (0.15% on GSM8K, 0% on HotpotQA). These results validate that the redundancy in the prefill stage is non-uniform and specifically concentrated in the deep layers, whereas shallow layers remain critical.

Necessity of Independent KV Projections. We evaluate a variant that removes the independent KV projections, and skips the KV cache generation for pruned layers entirely. With this variant, the model can only access the last input token and the generated tokens of the last 1/3 layers during the decode stage, while being unable to access initial input tokens. This results in catastrophic collapse in model accuracy (2.05% on GSM8K, 1.18% on HotpotQA), confirming that while the residual updates of deep layers are redundant for prefill, their KV states are indispensable for the full model to perform attention computations during the decode stage.

Importance of Boundary Handling. We evaluate a variant that removes the boundary handling for the last input token. With this variant, the x_N is also processed with the pruned prefill model. This variant suffers from obvious drop in accuracy on both tasks (80.21% to 77.33% on GSM8K, 63.13% to 11.45% on HotpotQA), indicating the necessity of processing the final token with the full model.

4.4.2 Sensitivity to Pruning Ratio

We further investigate the trade-off between inference efficiency and model performance by varying the pruning ratio from 20% to 60%. The prefill speedup is measured with a sequence length of 1024 and a batch size of 4. Experimental Results are presented in Table 4.

Table 4: **Impact of pruning ratio.** Performance and speedup trade-off at different pruning ratios on Qwen3-VL-8B-Instruct.

Pruning Ratio	**Speedup**	**GSM8K**	**HotpotQA**
0% (Full Model)	1.00×	81.50	65.49
20%	1.19×	83.09	65.46
25%	1.25×	82.34	65.81
33% (Default)	**1.37×**	**80.21**	**63.13**
40%	1.46×	80.82	61.69
50%	1.67×	78.54	34.69
60%	1.96×	38.51	5.45

We observe that at lower pruning ratios (20%-25%), the model maintains or even slightly surpasses the full model’s accuracy (e.g., 83.09% vs 81.50% on GSM8K). We hypothesize that mild pruning may act as a regularization mechanism, filtering out noise in the deep layers. However, these ratios offer limited speedup. Our default ratio of 33% achieves considerable acceleration (1.37×) with negligible accuracy loss. Pushing the ratio beyond 50% leads to a sharp decline in performance, particularly on HotpotQA, indicating that excessive pruning compromises the model’s capacity to encode complex context information.

5 Related Work

Model Pruning. Model pruning accelerates inference by removing redundant parameters. Unstructured pruning methods, such as SparseGPT (Frantar and Alistarh, 2023) and Wanda (Sun et al., 2024), prune individual weights based on magnitude and activation norms. While preserving accuracy, they often require specialized kernels to achieve wall-clock speedup. Structured pruning addresses this by removing coarse-grained components like layers or channels. Component-wise methods such as LLM-Pruner (Ma et al., 2023) and SliceGPT (Ashkboos et al., 2024) employ dependency graphs or matrix factorizations to prune structural units. In contrast, layer-wise methods like ShortGPT (Men et al., 2025), LaCo (Yang et al., 2024b) and SLEB (Song et al., 2024) demonstrate that specific layers in LLMs are redundant. However, existing structured pruning methods are typically stage-agnostic, applying the same reduced architecture across both prefill and decode stages. Our work challenges this paradigm by revealing that layer redundancy is highly stage-dependent, motivating a prefill-only pruning strategy.

Token Pruning and Compression. Complementary to parameter reduction, token pruning accelerates inference by reducing the sequence length. For text inputs, perplexity-based methods such as LLMLingua (Jiang et al., 2023; Pan et al., 2024) compress input length by selecting only the most informative tokens with a smaller model. In contrast, attention-based methods such as PyramidInfer (Yang et al., 2024a) and DAC (Zhao et al., 2025) determine token importance with attention weights. In the multimodal domain, token pruning methods such as FastV (Chen et al., 2024) and DART (Wen et al., 2025) mitigate the visual token redundancy by discarding or merging image tokens after several layers in the language model backbone, based on attention weights or token similarities. These methods can be applied along with our proposed POP.

Sparse Attention. Recent research also optimizes the attention mechanism to handle long contexts. For the compute-bound prefill stage, existing methods such as MInference (Jiang et al., 2024), MMInference (Li et al., 2025) and FlexPrefill (Lai et al., 2025) utilize block-sparse attention to bypass insignificant calculations. For the memory-bound decode stage, existing approaches such as Quest (Tang et al., 2024), PQCache (Zhang et al., 2025a) and MagicPIG (Chen et al., 2025) relieve the KV cache bottleneck by offloading KV cache to CPU memory, and perform sparse retrieval for computation. These methods can also be combined seamlessly with POP for further efficiency improvement.

6 Conclusion

In this work, we identify and exploit the asymmetric sensitivity to model pruning between the prefill and decode stages. Our analysis highlights that while deep layers are indispensable for generation (decode), they contribute minimally to context encoding (prefill). Based on this, we introduce POP, a simple yet effective strategy that accelerates the prefill stage by pruning deep layers, while preserving the full model for the decode stage. By decoupling the computational pathways of context processing and token generation, POP achieves prefill speedup of up to 1.37×, while maintaining the accuracy comparable to the full model, significantly outperforming existing structured pruning methods. Our findings suggest that stage-aware optimization is a promising direction for efficient LLM inference, potentially extending beyond pruning to other techniques such as quantization and model architecture design.

7 Limitations

While POP provides a compelling trade-off between efficiency and accuracy, we acknowledge several limitations.

First, unlike stage-agnostic pruning methods that permanently remove parameters to reduce memory footprint, POP requires the full model weights to be loaded for the decode stage. Consequently, it does not alleviate peak VRAM usage and is best suited for compute-bound rather than capacity-bound scenarios.

Second, our current implementation is based on a monolithic inference pipeline modified from the Transformers library. Recent advancements in inference systems such as DistServe (Zhong et al., 2024) and Splitwise (Patel et al., 2024) propose disaggregated systems that deploy prefill and decode instances on separate hardware resources. As POP naturally treats these two stages differently, it holds great potential for integration into these systems to further maximize cluster-level throughput. However, adapting POP for such distributed frameworks involves non-trivial engineering efforts, which we leave for future research.

Acknowledgments

We thank all the reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China (No. 62472330).

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 3119–3137. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXI*, volume 15139 of *Lecture Notes in Computer Science*, pages 19–35. Springer.

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Léon Bottou, Zhihao Jia, and Beidi Chen. 2025. Magicpig: LSH sampling for efficient LLM generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9313–9332. Association for Computational Linguistics.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13358–13376. Association for Computational Linguistics.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. 2019. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4158–4169.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. 2025. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yann LeCun, John S. Denker, and Sara A. Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann.

Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. 2025. Mminference: Accelerating pre-filling for long-context visual language models via modality-aware permutation sparse attention. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 20192–20204. Association for Computational Linguistics.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11264–11272. Computer Vision Foundation / IEEE.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 963–981. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient generative LLM inference using phase splitting. In *51st ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2024, Buenos Aires, Argentina, June 29 - July 3, 2024*, pages 118–132. IEEE.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards VQA models that can read. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8317–8326. Computer Vision Foundation / IEEE.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. SLEB: streamlining llms through redundancy verification and elimination of transformer blocks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Gemma Team. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. 2025. Stop looking for important tokens in multimodal language models: Duplication matters more. *CoRR*, abs/2502.11494.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Peng Wu. 2023. Pytorch 2.0: The journey to bringing compiler technologies to the core of pytorch (keynote). In *Proceedings of the 21st ACM/IEEE International Symposium on Code Generation and Optimization, CGO 2023, Montréal, QC, Canada, 25 February 2023- 1 March 2023*, page 1. ACM.

xAI. 2024. Realworldqa dataset. `https://huggingface.co/datasets/xai-org/RealworldQA`. Accessed: 2026-01-04.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024a. Pyramidinfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3258–3270. Association for Computational Linguistics.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 6401–6417. Association for Computational Linguistics.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, and 3 others. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 9556–9567. IEEE.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025a. Pqcache: Product quantization-based kvcache for long context LLM inference. *Proc. ACM Manag. Data*, 3(3):201:1–201:30.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025b. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 881–916. Association for Computational Linguistics.

Yi Zhao, Zuchao Li, Hai Zhao, Baoyuan Qi, and Liu Guoming. 2025. DAC: A dynamic attention-aware approach for task-agnostic prompt compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 19395–19407. Association for Computational Linguistics.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 193–210. USENIX Association.

A Robustness to Representation Mismatch

A potential concern regarding POP is the representation mismatch introduced by layer pruning. In the prefill stage, bypassing a layer l implies that the subsequent layer $l+1$ receives the input x_l directly, rather than the expected x_{l+1}. Since the deep layers were trained to process specific feature distributions, one might expect this mismatch to accumulate, corrupting the KV cache and leading to catastrophic collapse in the decode stage.

However, our approach addresses this risk through both theoretical safeguards and empirical verification:

Theoretical Safeguards via Virtual Gates. Theoretically, our importance estimation metric $\tilde{I}_l$ implicitly accounts for the sensitivity to representation mismatch. The score is derived from the gradient of the loss with respect to the virtual gate g_l:

$$\tilde{I}_l = \mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial g_l}\right)^2\right]$$

This gradient quantifies how much the final prediction loss $\mathcal{L}$ changes when layer l is removed. If skipping layer l leads to a severe distortion in subsequent layers, the gradient would exhibit large variance, resulting in a high importance score. Consequently, such layers would be retained by our strategy.

Empirical Verification with Functional Robustness. To understand the physical mechanism of this robustness, we conduct a layer-wise analysis on Qwen3-VL using the WizardLM-V2-196K dataset. Specifically, we measure the internal consistency between the pruned and full models within the deep 1/3 layers (layer 25-36). We track the cosine similarity for 3 key representations: (1) the hidden states of each layer; (2) the KV cache of input tokens; and (3) the attention outputs of the decode stage. As illustrated in Figure 2, we observe a striking contrast between representation drift and functional stability:

- **Representation Drift:** As expected, skipping layers accumulates numerical deviations. The similarity of hidden states gradually drops from 1.0 down to 0.71 (blue dashed line), and the value states in the KV cache (green dashed line) show even greater divergence, dropping to as low as 0.46. This confirms that the vector space of the pruned model indeed drifts from the original trajectory.

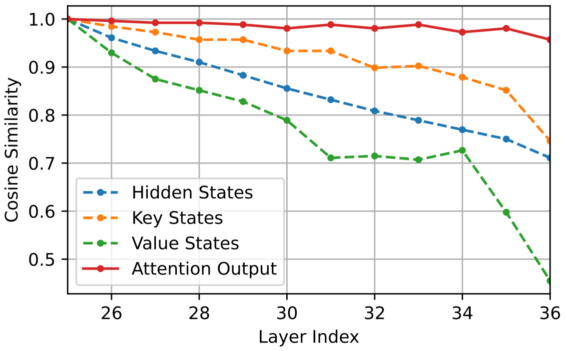


Figure 2: Cosine similarity of internal states between the POP-pruned model and the full model across deep layers (25-36).

- **Functional Stability:** Although the attention module receives drifted keys and values as inputs, its output maintains high fidelity. Specifically, the attention output (red solid line) consistently maintains a high cosine similarity, staying above 0.96 across all measured layers (25-36), significantly outperforming other internal states. This indicates that the attention mechanism acts as a robust stabilizer: the weighted aggregation over the context window effectively smooths out the noise from individual drifted tokens, ensuring the semantic information passed to the next layer remains valid.

Conclusion. Our analysis reveals that POP succeeds because the deep layers possess intrinsic functional robustness, where the attention mechanism compensates for the representation drift. Our virtual gate mechanism correctly captures this property: the low gradient variance calculated for these deep layers implies that the loss landscape is insensitive to the observed representation mismatch.

B Integration with Orthogonal Methods

As noted in our Related Work section (Section 5), POP (model pruning) is orthogonal to both sparse attention and token pruning methods. Technically, while token pruning reduces the sequence length N (temporal dimension) and sparse attention reduces computation within attention blocks (computational sparsity), POP physically removes entire transformer layers L (depth dimension). Thus, POP reduces the computational cost per token regardless of the sequence length or attention pattern, making it naturally complementary to these approaches.

To demonstrate this, we integrate POP with Flex-

Table 5: Accuracies on long context tasks and TTFT speedup ratios. Experiments are conducted with greedy decoding (Temperature = 0). TTFT speedup is measured with batch size 1, input length 32K.

Method	**TTFT Speedup**	**HotpotQA**	**MultiFieldQA**
Full Model	1.00×	57.83	56.00
FlexPrefill ($\gamma = 0.95$)	1.13×	56.39	54.41
FlexPrefill ($\gamma = 0.8$)	1.22×	47.79	48.55
FlexPrefill ($\gamma = 0.95$) + POP (31.25%)	**1.54**×	**54.34**	**53.42**
LLMLingua-2 (0.4)	1.37×	56.11	42.49
LLMLingua-2 (0.3)	1.53×	50.92	39.90
LLMLingua-2 (0.4) + POP (31.25%)	**1.56**×	**51.74**	**42.31**

Prefill (Lai et al., 2025) and LLMLingua-2 (Pan et al., 2024). Considering that both baselines are primarily designed to handle long-context challenges, we conducted additional experiments on Llama-3.1-8B-Instruct using the HotpotQA and MultiFieldQA datasets. Experimental results are shown in Table 5.

Experimental results demonstrate that **POP exhibits strong orthogonality with both sparse attention and token pruning methods**. Specifically, aggressively increasing sparsity in baselines (e.g., FlexPrefill $\gamma = 0.8$) yields limited speedup (1.22×) but suffers from significant accuracy drops (57.83 → 47.79 on HotpotQA). In contrast, combining a conservative setting of these methods (e.g., FlexPrefill $\gamma = 0.95$ or LLMLingua-2 ratio = 0.4) with POP achieves significantly higher speedups (∼1.56×) while preserving higher accuracies (54.34 / 51.74 on HotpotQA). This confirms that the joint application of POP with these orthogonal methods yields a superior Pareto frontier in terms of accuracy and efficiency.