# Budget-Aware Layer Pruning for Large Language Models

Anonymous ACL submission


## Abstract

Layer pruning (i.e., pruning whole Transformer layers) is attractive for large language model deployment because it yields real latency and throughput gains. The key challenge is selecting which layers to remove under a pruning budget $K$, where $K$ is the number of removed layers. Many methods reuse one global ranking for all budgets and then take the top-$K$ layers. This creates a nested-prefix constraint, meaning that if $K_1 \leq K_2$, the selected set for $K_1$ must be contained in the selected set for $K_2$. Through exhaustive search on LLaMA-3.1-8B, we show that optimal sets for different budgets can be non-nested, so one budget-independent ranking is insufficient. Motivated by this observation, we redefine layer scoring from intrinsic importance to a budget-aware continuation value and formulate pruning as a budget-conditioned finite-horizon Markov Decision Process. This formulation makes the learned value directly correspond to final pruned-model quality. During joint multi-budget training, we sample different budgets $K$ across episodes and optimize one shared scorer using transitions from all budgets; we approximate this value with Double Dueling DQN. Experiments show better quality than strong layer-pruning baselines under the same budget, stronger robustness under aggressive pruning, real wall-clock acceleration, and compatibility with post-training quantization. Our code is available at: `https://anonymous.4open.science/r/Budget-Aware-Layer-Pruning-B2CD`.


## 1 Introduction

Layer pruning is attractive for LLM deployment because it gives real wall-clock speedup and throughput gains (Kim et al., 2024; Men et al., 2024; Song et al., 2024; Ding et al., 2026). The key bottleneck is deciding which layers to remove under a given pruning budget. Existing methods usually reduce this decision to a single global layer ranking (Men et al., 2024; Kim et al., 2024; Ding et al., 2026) or iterative one-step deletion (Song et al., 2024).

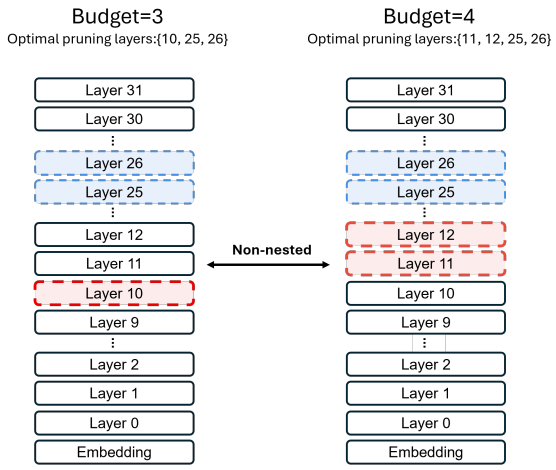



Figure 1: Optimal pruning sets on LLaMA-3.1-8B obtained by exhaustive search using the BookCorpus calibration set. The optima for $K = 3$ and $K = 4$ are non-nested, showing that optimal pruning sets can violate the nested-prefix structure imposed by fixed-ranking methods.

Here the pruning budget $K$ means how many layers we remove. This matters because pruning at small $K$ and large $K$ requires different trade-offs. In many existing methods, one global ordering is computed without explicit conditioning on $K$ (Men et al., 2024; Kim et al., 2024) and then reused for all budgets. Reusing one ordering $\pi$ gives $\mathcal{P}_K = \{\pi_1, \ldots, \pi_K\}$, which enforces a hidden *nested constraint* $\mathcal{P}_{K_1} \subseteq \mathcal{P}_{K_2}$ for $K_1 \leq K_2$. At the same time, many methods select layers by immediate local effect (Men et al., 2024; Kim et al., 2024; Song et al., 2024), without explicitly modeling how current choices change later choices.

Figure 1 shows why this is a real bottleneck. Through exhaustive search on LLaMA-3.1-8B for $K \in \{1, 2, 3, 4\}$, we find non-nested optima, $\mathcal{P}_3^* = \{10, 25, 26\}$ and $\mathcal{P}_4^* = \{11, 12, 25, 26\}$. No budget-independent ranking can represent both sets

simultaneously. This is the turning point of our method design. Layer scoring should depend on both pruning context and budget.

Based on this observation, we redefine layer scoring from intrinsic importance to a budget-aware continuation value. The score $q_\theta(s_t, a; K)$ should reflect not only immediate damage but also downstream pruning consequences under budget $K$. To give this continuation value a principled training target, we formulate pruning-set construction as a budget-conditioned finite-horizon MDP. Under a telescoping reward and $\gamma = 1$, the return equals the negative degradation of the final pruned model, so the optimal action-value function $Q^*(s_t, a; K)$ is exactly aligned with the pruning objective. This exact alignment, together with joint multi-budget training, allows one shared scorer to be learned and reused across pruning stages and budgets.

This formulation also motivates how we optimize the scorer in practice. At each step, the action is discrete (choose one layer), and we need to rank all feasible layers repeatedly under different budgets. A value-based method naturally provides this ranking signal and supports replay-based reuse across mixed-budget trajectories. We therefore use Double Dueling DQN(D3QN) (van Hasselt et al., 2016; Wang et al., 2016) to approximate this continuation value and generate the pruning set by deterministic rollout until $K$ layers are removed.

Experiments support this formulation with deployment-oriented gains. Under the same pruning budget, our method improves quality over competitive layer-pruning baselines and preserves downstream accuracy under aggressive pruning. It also converts layer reduction into real wall-clock acceleration (about $1.53\times$ on LLaMA-3.1-8B at $K$=12) and remains compatible with GPTQ-INT4 quantization, where quality is preserved while memory and latency gains compose.

The main contributions of this work are as follows:

- We provide exhaustive-search evidence that optimal layer-pruning sets can be non-nested across budgets. This finding reveals a structural limitation of budget-independent rankings, whose nested-prefix form cannot represent all optimal pruning sets.
- We propose a budget-aware contextual layer scorer that estimates continuation value rather than intrinsic importance. We formulate layer pruning as a budget-conditioned finite-horizon MDP with exact objective alignment, and use joint multi-budget training to learn one shared scorer across budgets. In implementation, we use Double Dueling DQN to approximate this score.
- We design a practical training setup that improves scorer reuse and stability across budgets, including mixed-budget replay and safety-constrained value updates. We then validate the full method on language modeling and zero-shot benchmarks, showing better quality under the same pruning budget, real wall-clock acceleration, and compatibility with post-training quantization.

## 2 Related Work

Existing layer-pruning approaches for LLMs mainly differ in how they define layer importance. Early methods often rely on local redundancy or sensitivity signals, such as hidden-state similarity in ShortGPT (Men et al., 2024), iterative redundancy estimation in SLEB (Song et al., 2024), and magnitude-, Taylor-, or perplexity-based criteria in Shortened-LLaMA (Kim et al., 2024). Other methods estimate sensitivity from training data, such as influence-function-based LayerIF (Askari et al., 2025). More recent work considers broader layer interactions, including search over pruning masks with calibration metrics in EvoPress (Sieberling et al., 2025), neighboring-layer redundancy in LaCO (Yang et al., 2024) and MKA (Liu et al., 2025), and coalition-level contributions estimated with Shapley-style values (Ding et al., 2026).

Overall, prior scoring rules draw on a broad range of signals, including representation change, parameter magnitude, output-distribution shift, objective sensitivity, and coalition-level contribution. Most of these scores are ultimately used to produce a budget-independent ranking: once a ranking or pruning order is obtained, different pruning counts are handled by truncating the same order.

## 3 Method

### 3.1 Problem Formulation

We formulate layer pruning (i.e., pruning whole Transformer layers) as budget-aware subset optimization and derive an exactly aligned sequential MDP as its learning surrogate. This formulation, combined with joint multi-budget optimization,

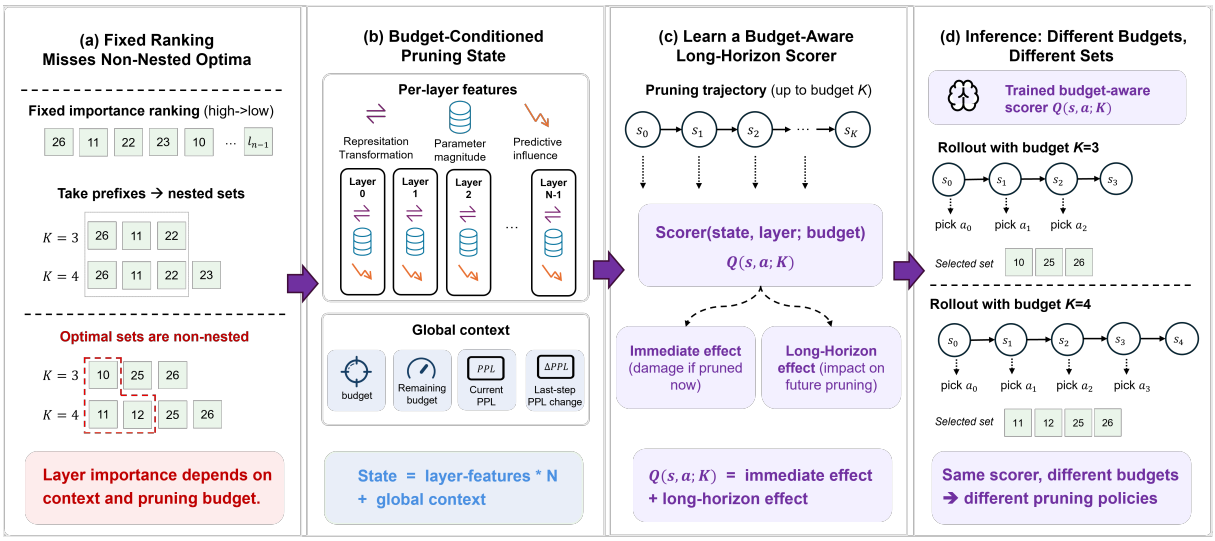



Figure 2: Overview of our budget-aware layer pruning framework. Unlike fixed-ranking methods, which take prefixes of a single pruning order and therefore impose nested pruning sets **(a)**, our method builds a budget-conditioned pruning state **(b)** and learns a long-horizon scorer $Q(s, a; K)$ **(c)**. The scorer evaluates each candidate layer under both the current pruning context and budget, enabling different, potentially non-nested pruning sets for different budgets at inference time **(d)**.

supports learning one shared scorer $q_\theta(s, a; K)$ across pruning stages and budgets.

### 3.1.1 General Formulation

Let $\mathcal{M}_0$ be a pre-trained LLM with $N$ Transformer layers $\{l_1, l_2, \ldots, l_N\}$. Let $\mathcal{D}(\mathcal{M}, \mathcal{M}_0)$ measure degradation relative to $\mathcal{M}_0$. Given budget $K$, we seek a layer set $\mathcal{P}$ with $|\mathcal{P}| = K$ that minimizes degradation:

$$\mathcal{P}^* = \arg \min_{\substack{\mathcal{P} \subseteq \{1,\ldots,N\} \\ |\mathcal{P}|=K}} \mathcal{D}(\mathcal{M}_0 \setminus \{l_i\}_{i \in \mathcal{P}}, \mathcal{M}_0)\,. \tag{1}$$

Layer pruning is not a fixed-ranking problem. The best layer to remove depends on budget $K$ and on jointly removed layers. Since the final model is determined by set $\mathcal{P}$ rather than removal order, pruning is a combinatorial subset-selection problem with search space $\binom{N}{K}$ (about $10^{10}$ for $N{=}36, K{=}18$). One-shot ranking is therefore structurally limited, because it assumes additive layer effects while LLM layers interact through residual and attention pathways.

To make this subset optimization learnable, we use a sequential computational form with horizon $K$. At step $t$, the policy observes state $s_t$ built from current model $\mathcal{M}_t$ and pruning context (§3.2), selects one feasible layer,

$$\begin{aligned} \mathcal{A}_t^{\text{feasible}} &= \mathcal{A}_t \cap \{\, i \mid M_{\text{safe}}(i) = 1 \,\}, \\ \mathcal{A}_t &= \{1, \ldots, N\} \setminus \mathcal{P}_{t-1}. \end{aligned} \tag{2}$$

and transitions deterministically to $\mathcal{M}_{t+1}$ after removing $l_{a_t}$. Here $M_{\text{safe}}$ is a static safety mask from one-pass pre-screening of catastrophic layers, shared by training and inference.

The reward uses telescoping degradation differences, $r_t = -\big[\mathcal{D}(\mathcal{M}_t, \mathcal{M}_0) - \mathcal{D}(\mathcal{M}_{t-1}, \mathcal{M}_0)\big]$, where $\mathcal{D}$ can be task-specific (e.g., log-perplexity, KL divergence, or accuracy loss). We set $\gamma = 1$ in this finite-horizon setting to preserve exact alignment with subset optimization.

**Set–Sequence Equivalence.** Under $\gamma = 1$ and the telescoping reward, the return collapses to

$$\begin{aligned} \sum_{t=1}^{K} r_t &= -\big[\mathcal{D}(\mathcal{M}_K, \mathcal{M}_0) - \mathcal{D}(\mathcal{M}_0, \mathcal{M}_0)\big] \\ &= -\mathcal{D}(\mathcal{M}_K, \mathcal{M}_0). \end{aligned} \tag{3}$$

Hence any two trajectories with the same final set $\mathcal{P}$ have identical total return. The sequential form is therefore an exact surrogate of the subset objective, not an order-biased heuristic. In contrast, $\gamma < 1$ down-weights late rewards and can favor specific orders even when the final subset is fixed.

### 3.1.2 Instantiation in This Work

We use $\mathcal{D} = \log \text{PPL}_{\mathcal{D}_{\text{cal}}}$ on held-out calibration data, which gives $r_t = -\log(\text{PPL}_t/\text{PPL}_{t-1})$ and is invariant to absolute perplexity scale. During training, we sample a pruning count $K$ from the budget set $\mathcal{K}$ per episode to realize the joint multi-budget objective discussed above.

### 3.2 State Design

Our goal is a compact state that is both Markovian and budget-aware. In implementation, we maintain a fixed per-layer feature tensor of shape $N \times d$ at every step and never change its shape during pruning. For each layer, we use a masked feature $\tilde{\mathbf{f}}_{i,t} = [m_{i,t}\mathbf{f}_{i,t}; m_{i,t}]$ with $m_{i,t} \in \{0, 1\}$, where $m_{i,t} = 1$ means layer $i$ is alive and $m_{i,t} = 0$ means layer $i$ is pruned. The full state is then

$$\begin{aligned} s_t &= [\tilde{\mathbf{f}}_{1,t}; \ldots; \tilde{\mathbf{f}}_{N,t}; \mathbf{m}_t], \\ s_t &\in \mathbb{R}^{N(d+1)+d_{\text{meta}}}. \end{aligned} \tag{4}$$

This slot-preserving design keeps input/output interfaces fixed, so one shared scorer can be trained across pruning steps and budgets rather than requiring budget-specific or step-specific networks.

**(i) Per-layer structural features.** For each layer $l_i$, we extract $\mathbf{f}_i \in \mathbb{R}^d$ with $d = 8$:

- **Representation transformation:** cosine similarity $\cos(\mathbf{h}_i^{\text{in}}, \mathbf{h}_i^{\text{out}})$, residual ratio $\|\mathbf{h}_i^{\text{out}} - \mathbf{h}_i^{\text{in}}\|/\|\mathbf{h}_i^{\text{in}}\|$, and output–input norm ratio $\|\mathbf{h}_i^{\text{out}}\|/\|\mathbf{h}_i^{\text{in}}\|$.
- **Parameter magnitude:** $L_1$ and $L_2$ norms of layer weights.
- **Predictive influence:** KL divergence between model outputs with/without $l_i$ on calibration tokens, plus entropy shift $\Delta H_i$.

Because all features are computed from the current model $\mathcal{M}_t$, transitions depend only on $(\mathcal{M}_t, a_t)$; the Markov property follows directly.

**(ii) Global context features.** $\mathbf{m}_t \in \mathbb{R}^{d_{\text{meta}}}$ with $d_{\text{meta}} = 4$ summarizes trajectory and budget: normalized progress $p_t = t/K$, normalized current perplexity $\widetilde{\text{PPL}}_t$, per-step perplexity change $\Delta\text{PPL}_t$, and budget ratio $K/N$. Including $K/N$ allows one policy to handle multiple budgets.

Per-layer features estimate each layer's current functional role, while global features disambiguate similar local patterns under different pruning stages and budgets. This combination is necessary for consistent decisions across multi-budget episodes, especially at larger budgets.

### 3.3 Budget-Aware Layer Scorer

Most ranking methods treat layer importance as intrinsic and budget-agnostic, then obtain different budgets by truncating one global ordering. We instead learn a budget-aware score under the current pruning state and directly use $q_\theta(s_t, a; K)$ as the score for feasible action $a \in \mathcal{A}_t^{\text{feasible}}$. Budget $K$ is encoded in $s_t$ through global context features, and we write $K$ explicitly to emphasize budget-aware conditioning. We use lowercase $q_\theta$ for the learned scorer and uppercase $Q^*$ for its optimal target. The score is therefore not intrinsic layer importance, but a budget-aware value under current state.

**Multi-Budget Joint Training.** During training, each episode samples a pruning count $K$ from the budget set $\mathcal{K}$, and a single shared network $q_\theta$ is optimized over transitions from all budgets rather than fitting separate per-budget maps. This coupling is the condition required for one scorer to produce comparable budget-aware scores across budgets. All transitions are written to a shared replay buffer and reused by off-policy updates, so data collected at different budgets jointly improve the same scorer. This directly increases sample utilization compared with budget-specific training.

We use a value-based formulation because the pruning action is discrete (choose one layer index), and at each step we need scores for all feasible layers under the current state and budget. A single forward pass of $q_\theta(s_t, \cdot; K)$ provides exactly this ranking signal, and replay over mixed-budget trajectories yields high sample efficiency through repeated cross-budget reuse. In implementation, Double Dueling DQN (van Hasselt et al., 2016; Wang et al., 2016) approximates $Q^*$ with $q_\theta : \mathcal{S} \times \mathcal{A} \times \mathcal{K} \to \mathbb{R}$ and $q_\theta(s_t, a; K) \approx Q^*(s_t, a; K)$, where $\mathcal{K} = \{K_{\min}, \ldots, K_{\max}\}$.

Double DQN reduces overestimation bias by decoupling action selection and target evaluation: the online network selects the best next feasible action, and the target network evaluates that action when forming the TD target. This makes $q_\theta(s_t, a_t; K)$ estimate long-term pruning effect rather than only one-step damage.

Because the objective is defined on the final subset, the score must capture downstream consequences of current action. Empirically, ablating budget conditioning degrades performance more under large $K$ (Fig. 8), supporting this formulation.

**Bellman View and Relation to Prior Methods.** From the Bellman view, each score combines immediate degradation change and the best feasible continuation value under the same budget. Selecting the highest-score action therefore optimizes both current and downstream pruning quality, which is crucial for path-dependent layer removal.

This view also clarifies prior methods: static-score approaches (Men et al., 2024) compute one ranking at $s_0$ and ignore state evolution, while iterative greedy methods (Song et al., 2024) re-score each step but optimize only immediate reward, effectively removing lookahead. With $\gamma = 1$, backup propagates full future cost to current decisions while preserving objective alignment.

Following the dueling architecture, $q_\theta$ is represented by a context-dependent state-value term plus a centered action-advantage term. This realizes budget-aware scoring, where the score of each candidate layer depends jointly on current context and budget.

**Inference.** At inference, generating a pruning set for budget $K$ is a deterministic rollout. We start from $s_0$, repeatedly remove the highest-scoring feasible layer under $q_\theta(s_t, \cdot; K)$, update state, and stop after $K$ steps. This yields $\mathcal{P}_K$. Because the scorer is budget-aware, the same trained network outputs different pruning sets for different $K$ without retraining.

### 3.4 Practical Training Design

To make the method practical to use, we design the training procedure as part of the method itself rather than as post-hoc heuristics. It supports shared optimization across budgets, robust state estimation with limited calibration compute, and stable value learning at high pruning ratios.

First, **Shared Multi-Budget Optimization.** We train with budget-mixed episodes and a shared replay buffer. This enforces one scorer across budgets at the optimization level and improves sample utilization by reusing experience collected at different $K$ for the same parameter set. Formally, we optimize one shared objective over mixed budgets, $\mathcal{L}(\theta) = \mathbb{E}_K \, \mathbb{E}_{(s,a,r,s') \sim \mathcal{B}_K} \big[ (y_t - q_\theta(s, a; K))^2 \big]$, where $\mathcal{B}_K$ denotes transitions collected under budget $K$, and all $\mathcal{B}_K$ are merged into one shared replay buffer in implementation.

Second, **Robust State Estimation.** We use stochastic calibration probing to construct state features. Instead of fixing one deterministic probe, we sample calibration slices during training so that the scorer is exposed to mild state-estimation noise and learns decisions that are less sensitive to a particular calibration draw. Let $\xi_t$ denote a sampled calibration slice at step $t$ with $\xi_t \sim p(\xi \mid \mathcal{D}_{\text{cal}})$ and $\mathbf{f}_{i,t} = \phi_i(\mathcal{M}_t; \xi_t)$, so the scorer is trained under a distribution of feature realizations rather than a single fixed estimate.

Third, **Safety-Constrained Value Learning.** We use safety-constrained value learning with feasible-action masking (Eq. 2), perplexity clipping, and catastrophic-step early termination. Together these constraints suppress outlier TD targets from pathological pruning actions and keep optimization in a numerically stable regime. Concretely, we clip rewards as $\tilde{r}_t = \text{clip}(r_t, r_{\min}, r_{\max})$, set $a^\star_{t+1} = \arg\max_{a \in \mathcal{A}^{\text{feasible}}_{t+1}} q_\theta(s_{t+1}, a; K)$, and use $y_t = \tilde{r}_t + \gamma \, q_{\bar{\theta}}(s_{t+1}, a^\star_{t+1}; K)$. We terminate an episode early if $\log \text{PPL}_{t+1} - \log \text{PPL}_t > \delta_{\text{stop}}$.

Detailed thresholds and implementation settings are reported in Section B.

## 4 Experiments

### 4.1 Experimental Setup

**Models** We evaluate our method on LLaMA3.1-8B (Grattafiori et al., 2024), Qwen3-8B and Qwen3-14B (Yang et al., 2025).

**Benchmarks.** We evaluate language modeling quality using perplexity on WikiText2 (Merity et al., 2016) and C4 (Raffel et al., 2023). We further evaluate zero-shot reasoning performance on PIQA (Bisk et al., 2019), HellaSwag (Zellers et al., 2019), ARC-easy and ARC-challenge(Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), RACE (Lai et al., 2017), WSC273 (Levesque et al., 2012), LAMBADA (Paperno et al., 2016), and MMLU (Hendrycks et al., 2021).

**Baselines.** We compare against layer pruning methods. Baselines include SLEB (Song et al., 2024), Shortened-LLM (Kim et al., 2024), and ShortGPT (Men et al., 2024).

**Implementation Details.** All experiments are implemented in PyTorch (Paszke et al., 2019) using HuggingFace Transformers (Wolf et al., 2020). For pruning calibration, we sample examples from BookCorpus and use the same calibration set across all baselines for a fair comparison.

We train the scorer with Double Dueling DQN using a lightweight calibration protocol. At each episode, we sample a small subset of calibration sequences from a fixed BookCorpus pool to compute the pruning state, which keeps per-step evaluation cost low while exposing the scorer to diverse calibration examples across episodes. The Q-network is a compact Transformer-based model that treats

| Model | Method | $K$=4 | | $K$=8 | | $K$=12 | | $K$=16 | |
|---|---|---|---|---|---|---|---|---|---|
| | | WikiText2 | C4 | WikiText2 | C4 | WikiText2 | C4 | WikiText2 | C4 |
| LLaMA3.1-8B | SLEB | 9.37 | 14.34 | 15.08 | 22.77 | 133.02 | 57.28 | 241.03 | 174.07 |
| | ShortGPT | 13.14 | 19.68 | 3437.19 | 4853.45 | 93462.21 | 16309.41 | 2127.54 | 1956.60 |
| | Shortened-LLM | 9.36 | **14.03** | 18.37 | 25.79 | 95.20 | 58.33 | 3173.21 | 1056.68 |
| | Ours | **8.68** | **14.03** | **14.27** | **21.57** | **33.87** | **36.50** | **170.71** | **133.43** |
| Qwen3-8B | SLEB | **10.40** | 17.50 | 18.09 | **23.77** | 35.82 | 47.47 | 84.56 | 119.05 |
| | ShortGPT | 29.43 | 31.32 | 494.82 | 232.54 | 1543.03 | 1373.41 | 10984.71 | 8457.11 |
| | Shortened-LLM | **10.40** | 17.50 | 18.47 | 28.43 | 44.27 | 58.92 | 119.59 | 156.81 |
| | Ours | 10.83 | **17.40** | **17.26** | 25.55 | **33.15** | **45.40** | **82.49** | **98.27** |
| Qwen3-14B | SLEB | **8.92** | **15.19** | 12.93 | **20.32** | 23.26 | 34.47 | 55.91 | 71.02 |
| | ShortGPT | 14.94 | 21.32 | 55.35 | 65.89 | 97.98 | 112.58 | 1960.00 | 1541.00 |
| | Shortened-LLM | 9.02 | 15.68 | 13.33 | 21.40 | 37.94 | 57.27 | 291.26 | 268.15 |
| | Ours | **8.92** | **15.19** | **12.87** | 20.41 | **22.92** | **33.37** | **55.00** | **69.64** |

Table 1: Perplexity (↓) on WikiText-2 and C4 under different pruning counts $K$ for LLaMA3.1-8B, Qwen3-8B and Qwen3-14B.

Transformer layers as tokens, prepends a global-context token, and outputs budget-conditioned layer scores through a dueling value-advantage head. We further condition the network on the pruning budget with a learnable budget embedding, enabling a single scorer to learn budget-specific pruning policies. Additional architecture and training details are provided in Appendix B.

### 4.2 Language Modeling

Table 1 compares our method with layer pruning baselines on WikiText2 and C4. Across LLaMA3.1-8B, Qwen3-8B, and Qwen3-14B, our method consistently achieves the lowest or near-lowest perplexity across pruning budgets. The advantage becomes more pronounced under aggressive pruning. In particular, on LLaMA3.1-8B, several baselines suffer sharp degradation when more layers are removed, whereas our method maintains substantially lower perplexity. These results indicate that our pruning strategy better preserves generative performance under large depth reduction.

We observed that ShortGPT becomes unstable under aggressive pruning on the evaluated model families. To reduce the possibility of implementation artifacts, we verified the official scoring rule, pruning order, calibration set, and the perplexity of the unpruned model. We therefore report capped or log-scale perplexity, together with downstream accuracy, to avoid drawing conclusions solely from extreme perplexity values.

### 4.3 Zero-shot Tasks

We compare layer pruning methods on Qwen3-8B across nine zero-shot multiple-choice benchmarks, as shown in Table 2. Overall, our method achieves the best or near-best average accuracy across all pruning budgets. At mild pruning levels, the advantage is consistent: when pruning four layers, our method obtains the highest average accuracy of 0.5714, compared with 0.5595 for both SLEB and Shortened-LLM. When pruning eight layers, our method again gives the best average accuracy, reaching 0.4826 and outperforming SLEB, Shortened-LLM, and ShortGPT, which obtain 0.4789, 0.4816, and 0.4001, respectively.

The advantage becomes more important under aggressive pruning. At $K = 12$, our method remains essentially tied with the best baseline, obtaining 0.4022 average accuracy compared with 0.4023 for SLEB, while clearly outperforming Shortened-LLM and ShortGPT. At $K = 16$, our method achieves the highest average accuracy of 0.3511, improving over SLEB by 1.91 points and over Shortened-LLM by 5.73 points. This suggests that the proposed target-aware scorer is more robust when a large number of layers must be removed.

Looking at individual tasks, our method preserves performance especially well on several reasoning and language understanding benchmarks. For example, at $K = 16$, it achieves the best scores on PIQA, HellaSwag, ARC-easy, ARC-challenge, RACE, LAMBADA, and average accuracy. It also substantially improves LAMBADA under aggressive pruning, obtaining 0.2111 at $K = 16$ compared with 0.1440 for SLEB and 0.0508 for Shortened-LLM. These results indicate that our

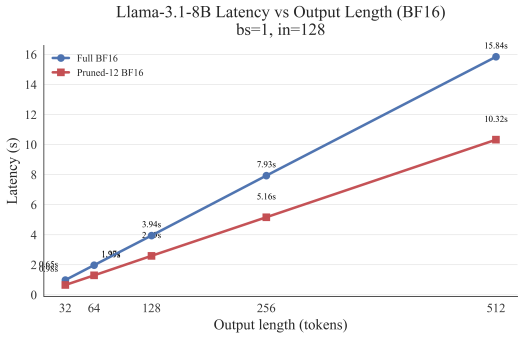
| K | Method | PIQA | HeSw | ARC-e | ARC-c | OBQA | Race | WSC273 | LAMBADA | MMLU | Avg acc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | SLEB | 0.7704 | 0.4849 | 0.7925 | 0.4898 | 0.2940 | 0.3923 | 0.7582 | 0.5030 | 0.5506 | 0.5595 |
| | ShortGPT | 0.6948 | 0.4737 | 0.6890 | 0.4326 | 0.2840 | 0.3703 | 0.7436 | 0.3194 | 0.6804 | 0.5209 |
| | Shortened-LLM | 0.7704 | 0.4849 | 0.7925 | 0.4898 | 0.2940 | 0.3923 | 0.7582 | 0.5030 | 0.5506 | 0.5595 |
| | Ours | 0.7650 | 0.5054 | 0.7832 | 0.4829 | 0.2940 | 0.3971 | 0.7436 | 0.5542 | 0.6175 | **0.5714** |
| 8 | SLEB | 0.7285 | 0.4293 | 0.7235 | 0.3831 | 0.2640 | 0.3474 | 0.6996 | 0.3513 | 0.3838 | 0.4789 |
| | ShortGPT | 0.6208 | 0.4039 | 0.4861 | 0.3038 | 0.2180 | 0.3282 | 0.5568 | 0.1556 | 0.5281 | 0.4001 |
| | Shortened-LLM | 0.7416 | 0.4494 | 0.7361 | 0.3951 | 0.2660 | 0.3531 | 0.6923 | 0.3450 | 0.3561 | 0.4816 |
| | Ours | 0.7296 | 0.4322 | 0.7226 | 0.4155 | 0.2760 | 0.3541 | 0.6703 | 0.3561 | 0.3873 | **0.4826** |
| 12 | SLEB | 0.6600 | 0.3653 | 0.6040 | 0.2969 | 0.2320 | 0.2957 | 0.6374 | 0.2435 | 0.2861 | **0.4023** |
| | ShortGPT | 0.6192 | 0.3724 | 0.4407 | 0.2782 | 0.2060 | 0.3120 | 0.5714 | 0.1003 | 0.3088 | 0.3565 |
| | Shortened-LLM | 0.6621 | 0.3618 | 0.6094 | 0.2628 | 0.2060 | 0.3091 | 0.6630 | 0.1877 | 0.2585 | 0.3912 |
| | Ours | 0.6600 | 0.3758 | 0.6006 | 0.2978 | 0.2460 | 0.3043 | 0.6227 | 0.2399 | 0.2726 | 0.4022 |
| 16 | SLEB | 0.5958 | 0.3026 | 0.4588 | 0.2278 | 0.2060 | 0.2517 | 0.5568 | 0.1440 | 0.2444 | 0.3320 |
| | ShortGPT | 0.5876 | 0.2771 | 0.3519 | 0.2628 | 0.1800 | 0.2287 | 0.5971 | 0.0019 | 0.3894 | 0.3196 |
| | Shortened-LLM | 0.5620 | 0.2917 | 0.3914 | 0.2056 | 0.1460 | 0.2488 | 0.5018 | 0.0508 | 0.2455 | 0.2938 |
| | Ours | 0.6164 | 0.3192 | 0.4634 | 0.2389 | 0.1880 | 0.2679 | 0.5934 | 0.2111 | 0.2617 | **0.3511** |

Table 2: Zero-shot accuracy comparison on Qwen3-8B across nine benchmarks under different pruning counts $K$.

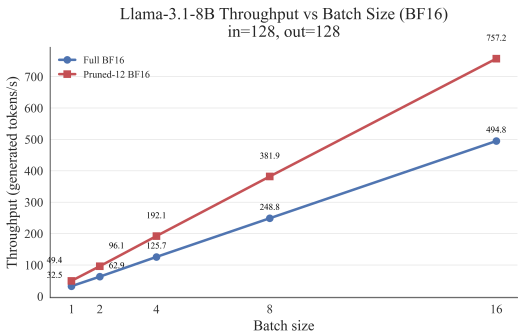
method better preserves layers that are important for downstream zero-shot generalization, especially in high-compression regimes.

### 4.4 Wall-Clock Inference Acceleration

We evaluate wall-clock acceleration on LLaMA3.1-8B ($N = 32$) by comparing the full BF16 model with our pruned model at $K = 12$ (Pruned-12), which retains 20 Transformer layers. As shown in Figure 3a, Pruned-12 consistently reduces single-stream generation latency by approximately $1.53\times$ across output lengths from 32 to 512 tokens, closely matching the depth-based upper bound of $N/(N-K) = 1.60\times$. Figure 3b further shows comparable gains in batched decoding throughput for batch sizes from 1 to 16. This consistency across both generation length and batch size suggests that the observed acceleration primarily comes from reduced per-token computation. These measurements confirm that the selected pruning configuration translates the structural reduction in model depth into practical latency and throughput improvements.




(a) Latency.



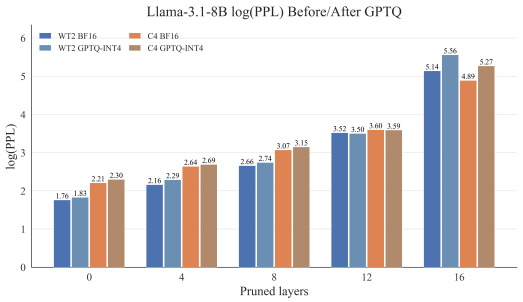

(b) Throughput.

Figure 3: Wall-clock inference acceleration of LLaMA3.1-8B after pruning 12 of its 32 layers.

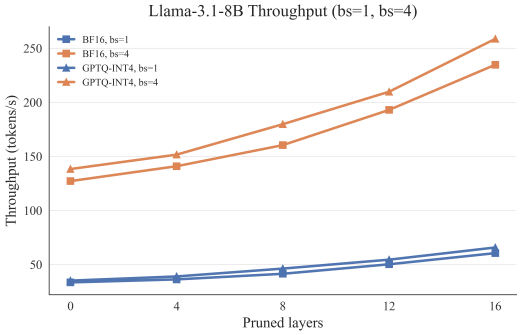
### 4.5 Compatibility with Post-Training Quantization

Layer pruning and weight quantization reduce model cost at different granularities, so their gains can be combined. Following Frantar et al. (2023), we apply GPTQ-INT4 to our pruned LLaMA-3.1-8B models with $K \in \{4, 8, 12, 16\}$ and evaluate perplexity and throughput.

As shown in Figure 4a, INT4 quantization introduces only a small perplexity increase over the corresponding BF16 pruned models across pruning counts. Figure 4b further shows that GPTQ-INT4 improves decoding throughput compared with BF16 under the same pruning count. These results indicate that our pruning method remains compatible with post-training quantization: language modeling quality is largely preserved, while throughput gains from reduced depth and low-bit weights can be combined.




(a) Log PPL.




(b) Throughput.

Figure 4: Compatibility of our layer pruning with GPTQ-INT4 on LLaMA3.1-8B across pruning budgets $K \in \{4, 8, 12, 16\}$.

### 4.6 Ablation Studies

We conduct two ablation studies to examine the main design choices in our pruning framework. The first analyzes how calibration data used for state construction affects the learned pruning scorer. The second evaluates whether explicitly conditioning the state on the pruning budget is necessary.

**Effect of calibration data for state construction.** In our framework, state features are computed by forwarding calibration samples through the current pruned model at each pruning step. The calibration configuration therefore affects both state quality and training cost. We study three factors: the calibration pool size, the number of sampled calibration examples per episode, and the sequence length of each calibration sample. The default setting uses a pool size of 64, 4 sampled examples per episode, and a sequence length of 512.

The full results are reported in Appendix C and Figures 5–7. Overall, the method is robust to moderate changes in calibration configuration, especially under mild pruning. Under larger pruning budgets, larger pools, more sampled examples, or longer sequences can improve performance in some cases, but the gains are not uniform across datasets and pruning budgets. These results suggest that the default calibration setting provides a reasonable trade-off between pruning quality and computational cost, and we use it in the main experiments unless otherwise stated.

**Effect of pruning budget conditioning.** Our MDP formulation includes the pruning budget as part of the state, allowing the scorer to evaluate each candidate layer under both the current pruning context and budget. To assess this design, we compare the full model with a variant that removes the pruning budget from the state while keeping all other components unchanged.

The full results are reported in Appendix D and Figures 8, showing that budget conditioning consistently improves pruning quality on both WikiText-2 and C4. The benefit is small under mild pruning but becomes larger as the pruning budget increases. This supports the motivation of our method: without budget information, the scorer must apply a single budget-agnostic criterion across all pruning budgets, which becomes inadequate under aggressive pruning. In contrast, budget conditioning allows the scorer to adapt its decisions to different pruning budgets and better model non-nested pruning behavior.

## 5 Conclusion

In this work, we revisit layer pruning for large language models from the perspective of budget-conditioned subset selection. We show through exhaustive search that optimal pruning sets can be non-nested across different pruning budgets, revealing a structural limitation of budget-independent ranking methods. Motivated by this observation, we propose a budget-aware contextual layer scorer and formulate the pruning process as a finite-horizon MDP. Under a telescoping reward with $\gamma = 1$, the MDP return is aligned with the final pruning degradation, and the optimal action-value function provides a principled long-horizon scoring target. We approximate this scorer with D3QN and use it to construct pruning sets conditioned on both the current pruning context and the pruning budget. Experiments on multiple LLM families and pruning budgets show that our method achieves lower perplexity and higher downstream task accuracy than strong depth-pruning baselines. These results suggest that explicitly modeling the pruning budget and context is important for effective layer pruning, and provide a practical framework for learning budget-aware pruning for LLM compression.

## Limitations

This work has several limitations. First, our method depends on calibration data to construct pruning states and compute rewards. Although our ablation study shows that the method is reasonably stable under different calibration settings, its performance may still vary with the quality, size, and domain coverage of the calibration set. Second, learning the D3QN scorer requires repeated forward passes through partially pruned models. This introduces additional training-time cost compared with static-ranking methods, although the learned scorer does not add extra cost to the final pruned model during inference. Third, our experiments focus on several LLaMA- and Qwen-family models and a fixed range of target pruning counts. Further evaluation on larger models, other architectures, and broader deployment settings is needed to better understand the generality of the proposed framework.

## References


Hadi Askari, Shivanshu Gupta, Fei Wang, Anshuman Chhabra, and Muhao Chen. 2025. LayerIF: Estimating layer quality for large language models using influence functions. In *Advances in Neural Information Processing Systems*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Xuan Ding, Pengyu Tong, Ranjie Duan, Yunjian Zhang, Rui Sun, and Yao Zhu. 2026. Pruning as a cooperative game: Surrogate-assisted layer contribution estimation for large language models. *Preprint*, arXiv:2602.07804.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened LLaMA: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *Preprint*, arXiv:1704.04683.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Deyuan Liu, Zhanyue Qin, Hairu Wang, Zhao Yang, Zecheng Wang, Fangying Rong, Qingbin Liu, Yanchao Hao, Xi Chen, Cunhang Fan, Zhao Lv, Zhiying Tu, Dianhui Chu, Bo Li, and Dianbo Sui. 2025. Pruning via merging: Compressing llms via manifold alignment based layer merging. *Preprint*, arXiv:2406.16330.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *Preprint*, arXiv:1809.02789.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

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 Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. 2025. Evopress: Accurate dynamic model compression via evolutionary search. *Preprint*, arXiv:2410.14649.

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 *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *Preprint*, arXiv:2402.11187.

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

# A The Use of Large Language Models

We used a large language model (LLM) only for editorial assistance, including grammar correction, phrasing refinement, and readability improvement. All substantive parts of this work, including the main ideas, method development, experimental design, implementation, and result analysis, were conducted independently by the authors. The LLM did not contribute to formulating the research questions, designing the methodology, running the experiments, or interpreting the findings.

# B Additional Training Details

**Calibration protocol.** We construct a calibration pool $\mathcal{C}$ containing 64 sequences of 512 tokens from BookCorpus. At the beginning of each episode, we uniformly sample $n_{\text{cal}} = 4$ sequences from $\mathcal{C}$ to compute the pruning state. This separates calibration diversity from per-step cost: each step evaluates only a small number of sequences, while different episodes observe different calibration subsets.

**Q-network architecture.** The Q-network treats Transformer layers as a sequence of tokens. Each layer is represented by an 8-dimensional feature vector and projected to hidden size $d = 128$. The 4-dimensional global context vector is projected to the same dimension and prepended as a `[CLS]` token. The resulting $(N+1)$-token sequence is fed into a 2-layer Transformer encoder with 4 attention heads and feed-forward dimension 512. We use a dueling head: the `[CLS]` token predicts the state value $V(s)$, while each layer token predicts its advantage $A(s, l_i)$. The final output is one Q-value for each layer action.

**Budget conditioning.** We use a learnable budget embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{K}| \times d}$, where $\mathcal{K}$ is the set of pruning budgets. Given a sampled budget $K$, the embedding $\mathbf{E}[K]$ is added to all token representations before the Transformer encoder:

$$\mathbf{h}_i \leftarrow \mathbf{h}_i + \mathbf{E}[K].$$

During training, budgets are sampled with probability $P(K) \propto K$, giving larger budgets more episodes because they have larger strategy spaces.

**RL Objective.** We train the scorer with Double Dueling DQN. For a transition $(s_t, a_t, r_t, s_{t+1}, d_t)$, the target is

$$y_t = r_t + \gamma Q_{\theta^-}\left(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a')\right)(1 - d_t),$$

where $\theta^-$ denotes the target network, $d_t$ is the terminal indicator, and $\gamma = 1.0$. The Q-network is optimized with mean squared error loss, and gradients are clipped to norm 10.0.

**Exploration.** We use $\epsilon$-greedy exploration with linear decay from 1.0 to 0.05 over 25K environment steps. Invalid actions, including already-pruned layers, are masked before action selection. We additionally use a warm-up stage of 100 iterations to collect transitions before gradient updates start.

**Training Configuration.** Unless otherwise specified, we optimize the Q-network with Adam using a learning rate of $10^{-4}$ and a batch size of 64. The replay buffer capacity is set to 25K transitions for multi-budget training. The target network is synchronized every 100 gradient steps, and each training iteration performs 50 gradient updates. We use 4 parallel GPU workers, with each worker collecting one episode per iteration. Training runs for 800 iterations, taking approximately 0.5 hour on 4×A100-80GB GPUs. Following common layer-pruning practice, we protect the first two layers and exclude layers $\{0, 1\}$ from pruning.

## C Calibration Configuration Ablation

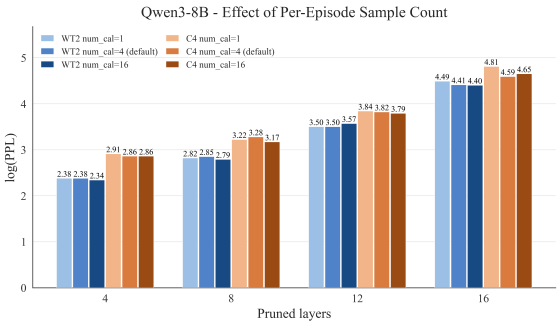



Figure 5: Ablation on the calibration number.

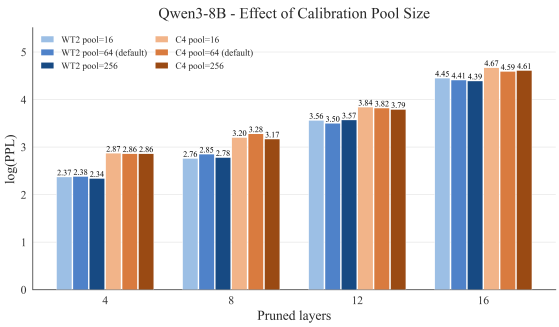



Figure 6: Ablation on the calibration pool size.

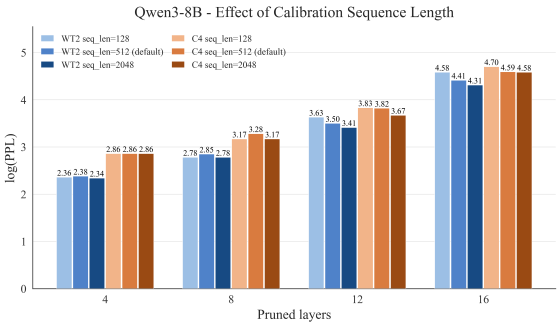



Figure 7: Ablation on the calibration sequence length.

## D Pruning budget Ablation

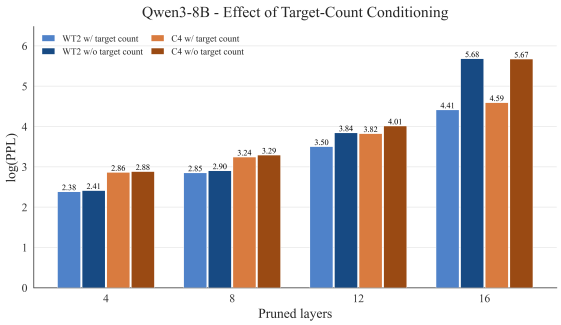



Figure 8: Effect of budget conditioning on Qwen3-8B layer pruning. For each pruning count $K$, we compare variants with and without budget information in the pruning state on WikiText-2 and C4. Conditioning on budget consistently reduces log perplexity, especially under larger pruning counts.