THINK HEAVY, ACT LIGHT: CONTEXT-AWARE HIERARCHICAL ACTIVATION SPARSITY FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation sparsity has emerged as a promising approach to reducing the inference cost of large language models (LLMs). However, existing post-training approaches face a quality–efficiency trade-off: input-agnostic sparsity budgets can degrade performance on challenging prompts, whereas learned token-level routing or online top-k/percentile selection introduce repeated decision and selection overhead during decoding. We introduce a hierarchical dynamic sparsification framework based on a *planning-and-execution* paradigm. During prefill, a context-aware reinforcement learning (RL) agent acts as a **Macro Controller** to produce prompt-conditioned layer-wise sparsity budgets, and a deterministic **Micro Router** greedily allocates these budgets across intra-layer modules. The corresponding module-specific magnitude thresholds are computed from the prefill activations and cached throughout generation. During autoregressive decoding, magnitude-mask generation is fused with the sparse computation kernel, eliminating per-token policy evaluation, greedy allocation, and percentile or top-k selection. All backbone parameters remain frozen throughout optimization; only a lightweight auxiliary actor–critic policy is trained. A separate policy is optimized for each backbone and target sparsity level without downstream-task supervision. Across five LLMs at 60%, 65%, and 75% sparsity, our method achieves the highest or tied-highest average downstream accuracy among the evaluated sparse methods in all 15 settings and the lowest or tied-lowest perplexity (PPL) in 29 of 30 settings. At 75% sparsity, it improves language-modeling perplexity and average downstream accuracy over the evaluated sparse baselines, although mathematical reasoning remains severely degraded relative to that of the dense models. The fused implementation provides low-overhead sparse decoding and achieves wall-clock speedups of up to $1.71\times$ over dense decoding on an NVIDIA A800 GPU.

1 INTRODUCTION

The capabilities of large language models (LLMs) come at a substantial computational cost, particularly during memory-bandwidth-bound autoregressive decoding. Activation sparsity can alleviate this bottleneck Zhang et al. (2022); Song et al. (2025; 2024). Because a substantial fraction of intermediate activations may contribute little to the output, activation sparsity can skip the corresponding computations Mirzadeh et al. (2023); Zhang et al. (2024); Li et al. (2022); Liu et al. (2023). Unlike conventional weight pruning Xia et al. (2023); Ma et al. (2023); Frantar and Alistarh (2023); Muralidharan et al. (2024), activation sparsity is applied dynamically, making it suitable for the diverse, heavy-tailed activation distributions of LLMs.

Despite its potential, current post-training activation sparsity methods often calibrate layer-wise schedules offline and reuse them across prompts. TEAL Liu et al. (2024) derives magnitude-based sparsity configurations through offline calibration, while WAS Wang et al. (2025) and LaRoSA Liu et al. (2025) account for depth-wise sensitivity through non-uniform scheduling or rotation but retain input-shared configurations. These designs do not model possible prompt-dependent variation in layer sensitivity. This motivates a complementary question: can a prompt-conditioned layer budget be constructed during prefill without adding a learned routing decision to every decoding step?

An effective sparsity strategy should adapt to the prompt without placing costly decision-making in the autoregressive loop. Evaluating a learned router or performing online top-k/percentile selection

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

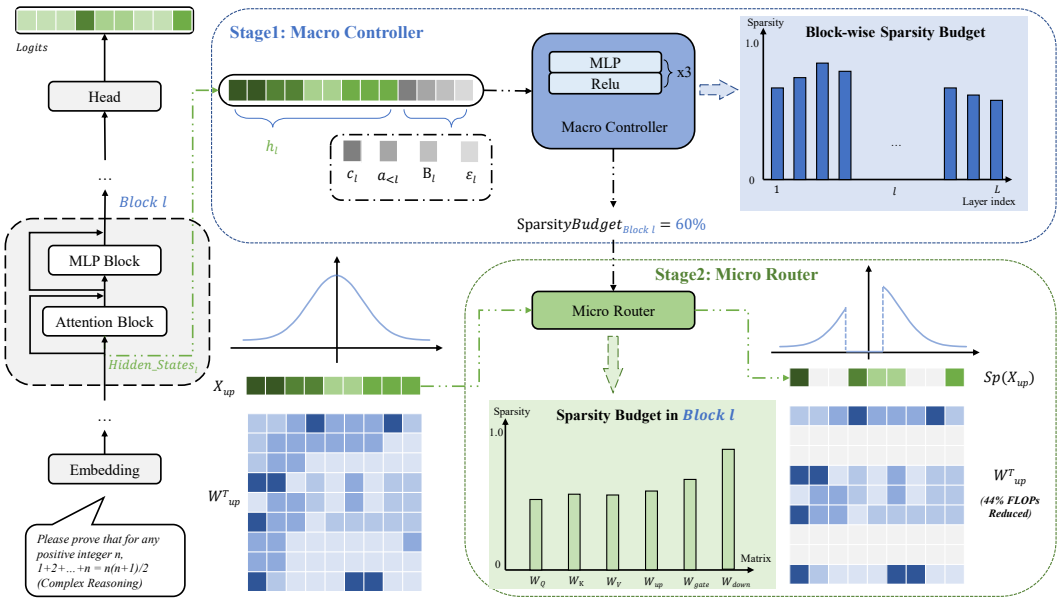

Figure 1: Our *planning-and-execution* framework. During prefill, the **Macro Controller** allocates prompt-conditioned layer-wise sparsity budgets, and the **Micro Router** distributes each budget across the linear modules and computes magnitude thresholds from the prefill activations. The budgets and thresholds are cached for generation. During autoregressive decoding, magnitude-mask generation is fused with sparse computation, avoiding repeated policy evaluation, greedy allocation, and percentile or top-k selection.

at every decoding step introduces repeated policy, selection, and kernel-launch overhead. Moreover, modeling fine-grained neuron-level selection (e.g., an action space of $2^{d_{\text{FFN}}}$ per feed-forward network (FFN), where d_{FFN} is the intermediate dimension) as an RL problem creates a prohibitively large stochastic action space and leads to high-variance policy optimization.

In this paper, we decouple the optimization problem into **Macro Budgeting** and **Micro Routing**. We introduce a context-aware reinforcement learning framework that schedules layer-wise sparsity from the prompt representation. The RL agent acts exclusively during *prefill*: the lightweight Macro Controller determines a prompt-conditioned set of layer-wise sparsity ratios, and the heuristic Micro Router distributes each assigned budget across the modules within that layer. The corresponding magnitude thresholds are then computed from the prompt's prefill activations and cached for generation. During decoding, token-dependent masks are generated inside fused sparse operators, following the kernel-integration strategy used by WAS Wang et al. (2025), while the policy, greedy allocator, and percentile operator remain outside the autoregressive loop. To stabilize RL training across heterogeneous prompts, we formulate the process as a finite-horizon Markov decision process (MDP) and introduce a dense-guided Kullback–Leibler (KL) divergence reward that separates policy quality from the intrinsic difficulty of different inputs.

To isolate the contribution of prompt conditioning, we independently train a matched prompt-free Macro Controller with the same dense-guided PPO Schulman et al. (2017) objective, data, optimization settings, Micro Router, sparsity budget, and decoder. This control removes all prompt-derived inputs from the actor and produces one fixed layer-budget trajectory shared across inputs during evaluation.

In summary, our main contributions are threefold:

- **Hierarchical Prefill-Time Planning:** We decouple scalar layer-budget optimization from deterministic module-level routing, compute prompt-conditioned budgets and thresholds during prefill, and reuse the cached thresholds in fused sparse decoding.

- **Dense-Guided Policy Optimization:** We formulate layer-wise allocation as an MDP and optimize a lightweight actor–critic policy with a dense–sparse KL reward while keeping the pretrained LLM frozen.
- **Matched Ablation and System Evaluation:** A matched, independently trained prompt-free control isolates the observed contribution of prompt inputs, while evaluations across five backbones and three sparsity levels assess accuracy and decoding efficiency. The fused implementation achieves up to $1.71\times$ speedup over dense decoding.

2 Related Work

Retraining-based sparsity. Retraining-based methods reshape activation distributions through activation-function changes or sparsity regularization during pretraining or fine-tuning. Replacing SwiGLU with ReLU can induce natural activation sparsity, while ReLU^2 further modifies the activation profile for efficient execution Mirzadeh et al. (2023); Zhang et al. (2024). ProSparse progressively increases sparsity regularization to preserve model quality at high sparsity Song et al. (2025). TurboSparse combines dReLU with sparsity-oriented data composition to obtain sparse activations Song et al. (2024). These approaches update the backbone, whereas our setting retains the pretrained LLM parameters.

Backbone-frozen sparsity. Backbone-frozen methods introduce activation sparsity without updating pretrained LLM weights. TEAL applies magnitude-based masking with a calibrated model-wide allocation Liu et al. (2024). Input-dependent selection has also been explored: CATS adjusts sparsity bounds using activation norms Lee et al. (2024), while DejaVu and predictor-based methods identify active FFN components from the current input Liu et al. (2023); Zhang et al. (2023). WAS combines constrained Bayesian optimization for inter-block allocation, greedy intra-block routing, and fused sparse kernels Wang et al. (2025). LaRoSA rotates activations to redistribute their magnitudes and improve robustness at high sparsity Liu et al. (2025). Our method optimizes a complete prompt-conditioned layer-budget trajectory once during prefill; deterministic micro routing then constructs module budgets and cached thresholds for fused decoding.

3 Methodology: Context-Aware Decoupled Dynamic Sparsity

The fundamental challenge of dynamic activation sparsity is to balance context-dependent representational capacity with hardware latency constraints. Let $X \sim \mathcal{D}$ denote an input prompt drawn from the data distribution and T the global target sparsity ratio. During prefill, our method constructs a prompt-conditioned configuration

$$\mathcal{C}(X) = \{\hat{p}_{l,m}(X), \hat{\tau}_{l,m}(X)\}_{l,m}\,, \tag{1}$$

where $\hat{p}_{l,m}(X)$ and $\hat{\tau}_{l,m}(X)$ are the allocated sparsity budget and cached magnitude threshold for module m in layer l, respectively, and m ranges over the set of linear modules $\mathcal{M}_l$. Let $f(X;\Theta,\mathcal{C}(X))$ represent the LLM with frozen weights Θ executed under this configuration. The pretrained LLM parameters remain frozen throughout policy optimization; only the macro controller and critic are updated. We minimize the task loss $\mathcal{L}$ subject to the allocated global sparsity budget:

$$\min_{\mathcal{C}(\cdot)} \mathbb{E}_{X\sim\mathcal{D}}\left[\mathcal{L}(f(X;\Theta,\mathcal{C}(X)))\right] \quad \text{s.t.} \quad \mathbb{E}_{X\sim\mathcal{D}}\left[\mathrm{Budget}(\mathcal{C}(X))\right] \geq T. \tag{2}$$

Here, $\mathrm{Budget}(\mathcal{C}(X)) = \frac{1}{L}\sum_{l=1}^{L}\sum_{m\in\mathcal{M}_l}\omega_{l,m}\hat{p}_{l,m}(X)$ is the parameter-footprint-weighted average of the allocated module sparsity ratios, where $\omega_{l,m}$ is the normalized parameter footprint of module m within layer l. Directly optimizing fine-grained neuron-level decisions at every generation step would create a prohibitively high-dimensional problem and introduce repeated routing overhead. As shown in Figure 1, our framework instead moves prompt-conditioned budget allocation and percentile-threshold computation to prefill. During **Hierarchical Budget Planning**, the Macro Controller first allocates layer-wise sparsity constraints, and the Micro Router then distributes each assigned budget across the modules within that layer and computes the corresponding magnitude thresholds. The resulting configuration $\mathcal{C}(X)$ is cached throughout generation. During **Autoregressive Decoding**, token-dependent magnitude masks are generated within fused sparse operators without invoking the policy, greedy allocator, or percentile or top-k computation.

3.1 MDP Formulation for Macro Budgeting

Directly optimizing fine-grained neuron-level routing couples inter-layer budget allocation with intra-layer component selection, resulting in a prohibitively high-dimensional decision problem. We therefore restrict the stochastic policy to a scalar layer-wise sparsity action at each step while delegating intra-layer allocation to a deterministic greedy router. This hierarchical decomposition reduces the stochastic decision space and separates macro-level policy learning from deterministic local allocation; it does not provide explicit layer-wise credit attribution. We model the resulting layer-wise budget allocation process as a finite-horizon Markov decision process (MDP), denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. The RL agent acts as the Macro Controller exclusively during prefill, computing a customized budget trajectory for subsequent decoding. Appendix A discusses this hierarchical decomposition.

State Space ($\mathcal{S}$): At layer $l \in \{1, \ldots, L\}$, the state s_l must encapsulate the semantic complexity of the prompt, the historical trajectory of routing decisions, and the remaining hardware budget. We extract the prompt context via an attention-aware pooling mechanism:

$$c_l = \mathrm{Softmax}\left(\frac{\mathbf{Q}_{\text{last}}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \in \mathbb{R}^{d_v}, \tag{3}$$

where $\mathbf{K}$ and $\mathbf{V}$ are the key and value matrices for all prompt tokens at layer l, d_k is the key dimension used for scaling, d_v is the value-head dimension, and $\mathbf{Q}_{\text{last}}$ is the query vector of the final prompt token.

The full state is defined as

$$s_l = [c_l \oplus u_l] \in \mathbb{R}^{d_s}, \qquad u_l = \mathrm{Enc}(h_l, \mathbf{a}_{<l}, B_l, \mathcal{E}(l)), \tag{4}$$

where u_l is a compact, fixed-width control-state representation constructed from the layer-local hidden representation h_l, the preceding sparsity decisions $\mathbf{a}_{<l}$, the remaining average sparsity budget B_l, and the sinusoidal layer-position encoding $\mathcal{E}(l)$. This representation retains both the layer-local information and the complete allocation history while providing a fixed-dimensional input to the policy network.

Action Space ($\mathcal{A}$): The policy network $\pi_\theta(a_l \mid s_l)$ parameterizes a normal distribution by predicting its mean μ_l and variance σ_l^2. To keep the action within a valid range, the agent uses truncated sampling to obtain a_l, which determines the pruning ratio for layer l. Here, $\mathcal{T}_l$ and $\mathcal{T}_r$ denote the left and right truncation boundaries, respectively.

$$\begin{aligned}(\mu_l, \sigma_l^2) &= \pi_\theta(s_l),\\ a_l &\sim \mathcal{TN}(\mu_l, \sigma_l^2, \mathcal{T}_l, \mathcal{T}_r).\end{aligned} \tag{5}$$

Transition Dynamics ($\mathcal{P}$): The state transition is governed deterministically by the global sparsity target T. After action a_l is taken, the average budget for the remaining layers is updated, for $l < L$, as

$$B_{l+1} = \frac{LT - \sum_{i=1}^{l} a_i}{L - l}. \tag{6}$$

This transition requires the agent to balance aggressive sparsity in shallow layers against lower sparsity in deeper layers.

Reward Function ($\mathcal{R}$): We formulate routing as a delayed-reward episodic task. For each intermediate decision ($l < L$), the immediate reward is zero ($r_l = 0$). At the terminal step L, the agent receives an episode reward R_L with two terms: a dense-guided KL term that measures how well the sparse model preserves the output distribution of the dense teacher and a budget-violation penalty for failing to meet the global sparsity target T. Section 3.2 defines R_L explicitly.

3.2 Dense-Guided Policy Optimization

Raw perplexity varies with prompt difficulty and therefore does not isolate the quality of a sparsity policy. We instead compare the sparse and dense next-token distributions on the same input. Let

$P_{\text{dense}}(y \mid X)$ and $P_{\text{sparse}}(y \mid X, \mathbf{a})$ denote these distributions for the dense model and the sparse model under trajectory $\mathbf{a}$, respectively. The constrained terminal reward is

$$R_L(X) = -\alpha \underbrace{\sum_{y \in \mathcal{V}} P_{\text{dense}}(y \mid X) \log \frac{P_{\text{dense}}(y \mid X)}{P_{\text{sparse}}(y \mid X, \mathbf{a})}}_{\text{knowledge-distillation penalty}} - \lambda \underbrace{\left(\max\left(0, T - \frac{1}{L} \sum_{l=1}^{L} a_l \right) \right)^2}_{\text{budget-violation penalty}}. \tag{7}$$

where $\mathcal{V}$ is the vocabulary, α scales the dense-guided KL term, and λ penalizes trajectories whose average sparsity falls below T. Additional pruning is disfavored when it increases the KL divergence. We optimize π_θ with standard clipped PPO: every layer action receives the same terminal return $R_L(X)$, and the critic $V_\phi(s_l)$ provides a state-dependent baseline. Appendix G gives the advantage and surrogate objectives together with the optimization settings.

3.3 Prefill-Time Micro Routing and Cached-Threshold Decoding

During prefill, the Macro Controller produces a prompt-conditioned trajectory $\mathbf{a}^* = [a_1^*, \ldots, a_L^*]$. The deterministic Micro Router distributes each a_l^* among the modules of layer l, reducing the remaining optimization to independent intra-layer allocation problems.

Prefill-Time Threshold Construction: Let $\mathcal{M}_l = \{W_q, W_k, W_v, W_o, W_{\text{up}}, W_{\text{gate}}, W_{\text{down}}\}$ and let $\mathbf{A}_{l,m}^{\text{pre}}(X)$ be the prefill activations of module m. After greedy allocation returns $\hat{p}_{l,m}(X)$, the prompt-specific threshold cached for decoding is

$$\hat{\tau}_{l,m}(X) = \text{Quantile}_{\hat{p}_{l,m}(X)} \left(\left| \mathbf{A}_{l,m}^{\text{pre}}(X) \right| \right). \tag{8}$$

Cached-Threshold Decoding: At decoding step t, let $\mathbf{x}_{l,m,t}$ denote the input activation of module m. The cached-threshold sparsification operator is

$$\left[S_{\hat{\tau}_{l,m}(X)}(\mathbf{x}_{l,m,t}) \right]_i = \begin{cases} x_{l,m,t,i}, & \text{if } |x_{l,m,t,i}| \geq \hat{\tau}_{l,m}(X), \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Activations and masks vary across decoding tokens, whereas the module budgets and thresholds remain fixed for each prompt. Following WAS Wang et al. (2025), threshold comparison, mask generation, and sparse linear computation are fused, avoiding online percentile or top-k selection.

For layer input $\mathbf{X}_l$, let $\mathbf{Y}_l = \text{Layer}_l(\mathbf{X}_l)$ and $\tilde{\mathbf{Y}}_l(\mathbf{X}_l; \mathbf{p}_l)$ denote the dense and sparse outputs under module sparsities $\mathbf{p}_l$, respectively. The constrained intra-layer target is

$$\mathbf{p}_l^{\text{opt}} \in \arg\min_{\mathbf{p}_l} \|\mathbf{Y}_l - \tilde{\mathbf{Y}}_l(\mathbf{X}_l; \mathbf{p}_l)\|_2^2 \quad \text{s.t.} \quad \sum_{m \in \mathcal{M}_l} \omega_{l,m} p_{l,m} = a_l^*, \tag{10}$$

where $\omega_{l,m} = \text{size}(W_{l,m}) / \sum_j \text{size}(W_{l,j})$ is the normalized parameter footprint.

Intra-Layer Greedy Search: We approximate this target by repeatedly increasing the sparsity of the module with the smallest marginal MSE and scaling its step inversely with its parameter footprint. The resulting allocation $\hat{\mathbf{p}}_l$ is not guaranteed to be globally optimal; Appendix B gives the complete algorithm.

4 Experiments

4.1 Datasets and Experimental Settings

Models, datasets, and baselines. We evaluate the framework across the open-source LLaMA family Touvron et al. (2023); Grattafiori et al. (2024), including LLaMA-2, LLaMA-3, and LLaMA-3.2, and the Mistral-7B Jiang et al. (2023) model. We measure language-modeling perplexity (PPL) on WikiText-2 Merity et al. (2016) and C4 Raffel et al. (2020). We also use the `lm-evaluation-harness` framework Sutawika et al. (2025) to evaluate knowledge retention, reasoning, and mathematical ability across nine downstream benchmarks: MMLU Hendrycks et al. (2020), C-Eval Huang et al. (2023), ARC Clark et al. (2018), GSM8K Cobbe et al. (2021), HellaSwag

Zellers et al. (2019), BBH Suzgun et al. (2023), WinoGrande Sakaguchi et al. (2021), PIQA Bisk et al. (2020), and BoolQ Clark et al. (2019).

We compare our approach with the post-training methods TEAL and LaRoSA. We reproduce TEAL using its original configurations. For LaRoSA, the reported sparsity coefficient α produced realized sparsity levels below the intended targets in our implementation. We therefore retain its best-performing coefficient while calibrating the experimental sparsity to the requested target. We also evaluate the $\alpha = 1$ configuration and report the better LaRoSA result in each setting. Because the public WAS release covers only a subset of the settings studied here, we report the available comparison in Appendix D.

Implementation Details.

We optimize the routing policies on $30{,}104$ contiguous $2{,}048$-token sequences from WikiText-103 while keeping all backbone parameters frozen. The actor and critic are separate three-layer MLPs with hidden dimension 256 and approximately 0.40M trainable parameters in total. We use AdamW with learning rate 10^{-4}, a batch size of 16 trajectories, and four PPO epochs per update. A separate policy is trained for each backbone–sparsity pair on eight Ascend 910B NPUs and is evaluated on WikiText-2, C4, and nine downstream tasks without adaptation. Decoding speed and throughput are measured at batch size 1 on one NVIDIA A800 GPU. Appendix G provides the remaining initialization, optimization, training-cost, and fused-kernel details.

Matched Prompt-Free Control. For the ablation in Table 3, we independently train a control policy after removing all prompt-dependent actor inputs, including the pooled context representation and layer-local hidden representation. It retains only the prompt-independent control state and uses the same training data, dense-guided PPO objective, hyperparameters, Micro Router, target budget, and fused decoder as the prompt-conditioned policy. During evaluation, it produces one fixed learned layer-budget trajectory shared across all prompts. "Prompt-free" therefore refers only to the Macro Controller; the activation-dependent Micro Router is unchanged.

4.2 Main Results

4.2.1 Language Modeling

We evaluate perplexity (PPL) on WikiText-2 and C4 (Table 1) to assess the retention of language-modeling ability. Among the evaluated sparse methods, our method achieves the lowest PPL in 28 of the 30 settings and ties TEAL in one additional setting: LLaMA-2-13B on C4 at 60% sparsity. The only setting in which it does not achieve the best PPL is LLaMA-2-7B on C4 at 60% sparsity, where LaRoSA performs best. The largest improvements occur at 75% sparsity. On C4 with LLaMA-3-8B, for example, the PPL values of TEAL and LaRoSA rise to 84.39 and 89.35, respectively, whereas our complete framework obtains 36.32. This result shows that the full method is more robust at high sparsity within the evaluated baseline set. The degradation under extreme sparsity is particularly large for the LLaMA-3 family. On WikiText-2 at 75% sparsity, our method achieves a PPL of 46.64 on LLaMA-3.2-3B, compared with 126.27 for TEAL.

4.2.2 Downstream Tasks

We assess the effect of dynamic sparsity on commonsense reasoning, reading comprehension, and mathematics across nine benchmarks (Table 2).

Strong Performance in Moderate Sparsity Regimes. At 60% and 65% sparsity, our framework improves upon or matches the strongest sparse baseline in average accuracy across the evaluated backbones. For LLaMA-2-7B at 60% sparsity, our method achieves an average accuracy of 50.75, improving over TEAL (50.26) by 0.49 percentage points, while remaining 5.28 points below the dense baseline (56.03). Thus, the full framework preserves more downstream accuracy than the evaluated sparse baselines in this setting.

Graceful Degradation at High Sparsity (75%). The advantage of our approach is most pronounced at 75% sparsity, where the evaluated baselines experience large accuracy losses. On LLaMA-3-8B, TEAL and LaRoSA obtain average scores of 37.23 and 38.23, respectively, while our method achieves 41.43 (+4.20 over TEAL). Our framework therefore reduces the loss in average downstream accuracy relative to the evaluated sparse baselines.

Table 1: Language modeling performance (perplexity; lower is better) on WikiText-2 and C4 across five LLMs at 60%, 65%, and 75% sparsity. Dense models are included as references. Our full framework achieves the best or tied-best result in 29 of the 30 sparse settings.

Dataset	Sparsity	Method / Model	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3.2-3B	LLaMA-3-8B	Mistral-7B
WikiText-2	0%	Baseline	4.87	4.42	6.98	5.61	4.76
	60%	TEAL	5.79	4.99	9.56	7.51	5.42
		LaRoSA	5.89	5.21	10.42	7.76	5.90
		Ours	**5.76**	**4.98**	**9.50**	**7.42**	**5.37**
	65%	TEAL	6.53	5.39	12.55	8.90	5.90
		LaRoSA	6.72	5.84	13.54	9.43	6.89
		Ours	**6.46**	**5.36**	**11.71**	**8.86**	**5.81**
	75%	TEAL	32.35	9.19	126.27	41.50	10.58
		LaRoSA	13.43	9.88	48.90	30.53	15.04
		Ours	**11.80**	**7.58**	**46.64**	**19.36**	**8.73**
C4	0%	Baseline	6.67	6.16	10.22	8.48	7.52
	60%	TEAL	8.08	**7.03**	15.09	12.27	8.51
		LaRoSA	**7.78**	7.22	16.44	12.47	8.93
		Ours	8.07	**7.03**	**14.87**	**12.17**	**8.42**
	65%	TEAL	9.29	7.66	20.14	14.65	9.19
		LaRoSA	9.31	7.97	22.71	15.34	10.17
		Ours	**9.27**	**7.62**	**18.59**	**14.49**	**9.06**
	75%	TEAL	61.43	13.65	153.51	84.39	16.96
		LaRoSA	21.37	14.38	190.09	89.35	25.45
		Ours	**17.83**	**10.88**	**83.48**	**36.32**	**13.44**

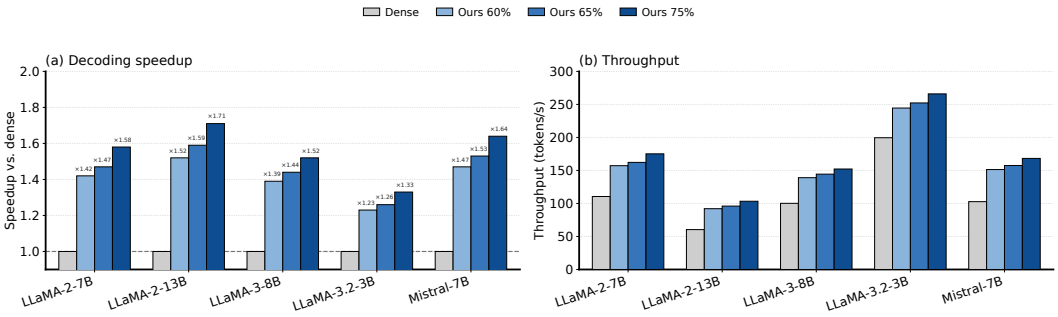

Figure 2: Autoregressive decoding efficiency on one NVIDIA A800 GPU at batch size 1: (a) speedup over dense decoding and (b) throughput. Measurements include threshold comparison, fused mask generation, and sparse linear computation, but exclude prefill-time planning.

Task-Specific Performance. Improvements vary across tasks. At 75% sparsity, our method improves MMLU on LLaMA-2-13B from 29.42 for TEAL to 34.53, whereas gains on GSM8K are less consistent across models and sparsity levels. For example, at 75% sparsity, our method scores 1.52 on GSM8K with LLaMA-3-8B, compared with 1.74 for TEAL. These results show that improvements in average downstream accuracy do not imply uniform gains on every task.

4.3 Computational Cost and Inference Efficiency

Figure 2 reports autoregressive decoding speedup and throughput relative to the dense baseline for all five models at 60%, 65%, and 75% sparsity, measured with a batch size of 1 on one NVIDIA A800 GPU. The measurements include threshold comparison, fused mask generation, and sparse linear computation; prompt-conditioned allocation and threshold construction occur during prefill and are excluded. The Macro Controller adds approximately 0.5% FLOPs relative to dense prefill. Following WAS Wang et al. (2025), mask generation is fused with sparse computation, and the actor, Micro Router, and percentile computation are absent from the decoding loop. Speedups range from $1.23\times$–$1.52\times$ at 60% sparsity and from $1.33\times$–$1.71\times$ at 75%, with the maximum observed for LLaMA-2-13B. Throughput increases monotonically with sparsity for every model.

Table 2: Zero-shot and few-shot downstream performance across five LLMs at 60%, 65%, and 75% activation sparsity. Our full framework achieves the highest or tied-highest average accuracy among the evaluated sparse methods in all 15 model–sparsity settings, with the largest improvements over the strongest sparse baseline at 75% sparsity.

Model	Sparsity	Method	MMLU 5-shot	C-Eval 5-shot	ARC-E 25-shot	GSM8K 5-shot	HellaSwag 10-shot	BBH 3-shot	WinoGrande 5-shot	PIQA 0-shot	BoolQ 32-shot	Avg.
LLaMA-2-7B	0%	Baseline	45.68	33.28	80.68	14.03	58.92	38.83	74.66	77.80	80.40	56.03
	60%	TEAL	36.87	27.27	76.98	7.81	53.89	30.36	68.90	76.17	74.07	50.26
		LaRoSA	34.09	27.27	77.02	5.53	53.04	30.30	65.75	76.06	64.98	48.23
		Ours	38.48	29.05	77.74	6.90	54.45	29.84	68.98	76.77	74.53	**50.75**
	65%	TEAL	32.05	26.75	75.25	4.47	50.33	29.53	65.04	75.84	69.24	47.61
		LaRoSA	33.24	26.15	73.65	4.85	48.39	28.95	63.38	75.79	68.69	47.01
		Ours	33.86	25.71	75.76	4.09	51.17	29.18	65.11	76.12	71.65	**48.07**
	75%	TEAL	25.75	27.49	53.24	0.99	30.43	26.06	52.57	71.33	60.64	38.72
		LaRoSA	26.65	24.81	59.09	1.67	35.12	27.55	53.99	72.25	61.56	40.30
		Ours	26.34	26.30	64.64	1.36	41.69	27.60	58.96	71.55	65.23	**42.63**
LLaMA-2-13B	0%	Baseline	55.19	37.74	82.87	22.90	61.69	39.16	76.64	79.05	84.25	59.94
	60%	TEAL	47.86	32.84	80.77	16.30	59.13	34.97	72.93	78.35	81.44	56.07
		LaRoSA	48.23	33.06	79.71	13.04	56.82	32.48	72.14	76.82	70.55	53.65
		Ours	48.86	34.55	81.06	15.69	59.18	34.45	73.48	77.31	80.73	**56.13**
	65%	TEAL	43.82	32.17	79.38	10.24	57.15	33.16	69.61	78.02	78.32	53.54
		LaRoSA	44.14	31.95	78.03	9.10	53.66	30.73	68.82	76.99	65.47	50.99
		Ours	45.30	31.50	79.41	9.62	57.54	31.73	71.03	76.61	79.17	**53.55**
	75%	TEAL	29.42	26.89	70.83	0.99	45.41	29.58	58.17	74.37	66.64	44.70
		LaRoSA	31.55	27.27	65.95	1.97	41.24	27.46	58.72	73.83	62.20	43.35
		Ours	34.53	27.56	73.53	2.81	49.93	29.78	63.38	73.55	72.87	**47.55**
LLaMA-3-8B	0%	Baseline	65.25	52.08	84.34	48.59	61.87	44.30	76.64	79.65	83.15	66.21
	60%	TEAL	52.62	39.67	78.45	19.41	53.93	37.47	68.11	77.20	75.44	55.81
		LaRoSA	51.86	38.71	77.10	18.95	52.58	36.14	67.65	76.61	72.29	54.64
		Ours	53.29	41.08	78.41	19.33	53.89	37.67	71.82	76.66	75.72	**56.43**
	65%	TEAL	44.92	34.25	75.00	11.98	50.17	33.78	65.98	75.46	70.24	51.31
		LaRoSA	43.63	34.84	73.44	10.01	48.04	33.13	63.22	76.44	66.82	49.95
		Ours	46.25	37.52	75.72	10.77	49.96	33.93	65.43	75.30	70.09	**51.66**
	75%	TEAL	25.71	25.63	48.40	1.74	30.88	27.51	52.01	71.22	52.01	37.23
		LaRoSA	26.56	26.45	48.53	0.83	30.61	23.88	53.43	72.20	61.59	38.23
		Ours	31.18	24.41	60.02	1.52	39.45	28.85	56.75	69.59	60.06	**41.43**
LLaMA-3.2-3B	0%	Baseline	56.43	45.25	78.62	26.08	56.72	37.15	72.61	76.66	76.85	58.45
	60%	TEAL	41.32	34.25	73.27	10.16	48.66	31.36	63.06	73.50	67.55	**49.24**
		LaRoSA	42.07	34.10	72.81	8.49	47.06	31.73	62.27	72.85	67.71	48.79
		Ours	41.92	34.10	72.64	10.61	48.12	30.70	62.75	73.56	68.72	**49.24**
	65%	TEAL	33.16	29.20	69.19	4.02	42.89	29.80	59.67	72.58	64.53	45.00
		LaRoSA	33.46	28.23	68.35	3.71	42.31	29.43	56.51	71.11	64.74	44.21
		Ours	34.70	30.76	68.98	4.55	43.27	29.64	59.19	72.20	66.61	**45.54**
	75%	TEAL	25.43	26.52	38.76	1.21	29.33	26.34	49.88	62.13	53.30	34.77
		LaRoSA	25.20	27.71	45.03	1.29	27.56	25.66	51.78	66.10	62.02	36.93
		Ours	27.50	26.30	51.26	1.14	33.00	27.35	54.70	66.16	61.68	**38.79**
Mistral-7B	0%	Baseline	62.47	44.06	84.47	34.95	62.42	43.48	77.74	80.30	85.35	63.92
	60%	TEAL	55.78	38.71	82.11	23.88	58.73	38.03	73.80	78.56	82.63	59.14
		LaRoSA	53.60	38.04	80.09	19.33	55.66	35.75	70.96	77.86	69.60	55.77
		Ours	55.54	39.30	81.90	23.43	58.71	37.95	74.59	78.67	82.60	**59.19**
	65%	TEAL	51.25	36.85	80.30	17.51	56.94	35.28	70.09	78.62	79.30	56.24
		LaRoSA	46.85	33.95	77.57	11.07	51.72	33.27	67.40	77.09	64.77	51.52
		Ours	51.14	35.44	80.47	18.20	56.42	36.00	70.72	78.56	80.03	**56.33**
	75%	TEAL	32.13	26.52	71.21	2.12	45.41	30.13	62.67	75.95	63.82	45.55
		LaRoSA	27.95	27.49	56.78	1.14	36.84	27.92	56.59	72.96	62.20	41.10
		Ours	37.61	27.93	74.03	3.87	47.40	31.45	63.77	76.33	65.90	**47.59**

4.4 Ablation Study

Table 3 separates the dense-guided objective from prompt conditioning on LLaMA-2-7B at 75% sparsity. Replacing naive policy gradient Sutton et al. (1999) with prompt-free dense-guided PPO reduces WikiText-2 PPL from 13.59 to 11.68 and raises Avg._9 from 40.45 to 41.04.

The second and third rows provide the matched comparison for prompt conditioning. Both policies are independently trained with the same data, objective, hyperparameters, Micro Router, budget, and decoder. The prompt-free variant removes all prompt-dependent actor inputs, including the pooled context and layer-local hidden representations, and uses one fixed learned macro trajectory for every evaluation prompt. Restoring these prompt features raises Avg._9 from 41.04 to 42.63, while PPL changes from 11.68 to 11.80. The observed contribution of prompt-conditioned macro budgeting in this setting is therefore $+1.59$ percentage points in average downstream accuracy, accompanied by a 0.12 increase in PPL. Because this comparison covers one backbone, one sparsity

Table 3: Matched ablation on LLaMA-2-7B at 75% sparsity. The prompt-free and prompt-conditioned policies are trained independently with the same data, dense-guided PPO objective, optimization settings, Micro Router, target budget, and fused decoder; they differ only in whether the Macro Controller receives prompt-dependent features. Prompt-free applies only to the Macro Controller, while the activation-dependent Micro Router remains unchanged.

Policy Variant	**PPL**$_{\text{WikiText-2}}$ $\downarrow$	**Avg.**$_9$ $\uparrow$
Naive policy gradient	13.59	40.45
Dense-guided PPO, prompt-free fixed schedule	**11.68**	41.04
Dense-guided PPO, prompt-conditioned	11.80	**42.63**

target, and one policy run, it should not be interpreted as a universal effect size. Appendix E provides a complementary post-hoc analysis of task-derived schedule transfer.

5 Conclusion

In this work, we introduced a hierarchical dynamic sparsification framework that balances context-aware capacity allocation with inference efficiency in Large language models. By decoupling routing optimization into a Reinforcement-learning-driven Macro Budgeting stage and a deterministic heuristic Micro Routing stage, our method completes prompt-conditioned budget allocation and threshold computation during prefill. Autoregressive decoding therefore requires no policy evaluation, greedy allocation, or percentile selection. Following the fused-kernel strategy of prior activation-sparsity systems, token-dependent magnitude-mask generation is integrated directly into sparse computation, enabling low-overhead decoding while preserving prompt-conditioned allocation.

Extensive evaluations across diverse LLM families (e.g., LLaMA and Mistral) demonstrate that our framework improves the empirical accuracy–efficiency frontier of activation sparsity. At 75% sparsity, our method improves language-modeling perplexity and average downstream accuracy over the evaluated sparse baselines, although mathematical reasoning remains severely degraded relative to the dense models. In the matched LLaMA-2-7B ablation at 75% sparsity, adding prompt-dependent inputs to the independently trained Macro Controller raises the observed nine-task average by 1.59 percentage points over the prompt-free control, while PPL increases by 0.12. On an NVIDIA A800 GPU, the resulting fused sparse implementation yields up to $1.71\times$ decoding speedup over dense inference.

AI USE STATEMENT

In this work, we used generative AI tools primarily for language polishing, literature discovery, consistency checking, and adversarial review of the manuscript. The research methodology, mathematical formulation, experimental design, implementation, analysis, and conclusions were developed and verified by the authors. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical correctness, consistency with the implementation and recorded experimental results, and faithful representation of our contributions. Generative AI was not used to create, modify, or fabricate experimental measurements. We take full responsibility for the final content of this work, including all claims, citations, figures, tables, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies post-training activation sparsity for reducing the inference cost of large language models. Our experiments use publicly available pretrained models, text corpora, and established evaluation benchmarks, and do not involve human subjects, user studies, or the collection of private or sensitive personal information. The proposed method is intended to reduce the computational and memory costs of deploying language models. Improved inference efficiency may lower deployment barriers, but it may also make the existing capabilities and associated risks of the underlying models more accessible. Moreover, sparsification may affect different tasks or capabilities unevenly, and therefore compressed models should be evaluated under the safety requirements appropriate to their intended applications. Our method does not introduce new training data or capabilities beyond those of the original models. The limitations, licenses, biases, and potential misuse risks of the underlying models and datasets remain applicable. We take responsibility for the ethical considerations associated with the methodology, experiments, and reported results.

REPRODUCIBILITY STATEMENT

We provide the policy architecture, state and action definitions, reward formulation, optimization hyperparameters, training-data composition, evaluation settings, and hardware information in the main paper and appendix. The appendix additionally reports the Micro Router pseudocode, auxiliary-policy training cost, and the boundary of the reported latency measurements. The pretrained models, training datasets, and evaluation benchmarks used in our experiments are publicly available. Baseline methods are reproduced using their official implementations and configurations whenever available; the calibration and implementation differences described in the paper are evaluated under matched sparsity and evaluation settings.

REFERENCES

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, 2022.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, et al. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2626–2644, 2025.

Yixin Song, Haotong Xie, Zhengyan Zhang, Bo Wen, Li Ma, Zeyu Mi, and Haibo Chen. Turbo sparse: Achieving llm sota performance with minimal activated parameters. *arXiv preprint arXiv:2406.05955*, 2024.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. Relu^2 wins: Discovering efficient activation functions for sparse llms, 2024. URL https://arxiv.org/abs/2402.03804.

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

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

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

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR, 2023.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102, 2024.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. *arXiv preprint arXiv:2408.14690*, 2024.

Ming Wang, Miao Zhang, Xuebo Liu, and Liqiang Nie. Weight-aware activation sparsity with constrained bayesian optimization scheduling for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1086–1098, 2025.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and Lulu Hu. La rosa: Enhancing llm efficiency via layerwise rotated sparse activation. *arXiv preprint arXiv:2507.01299*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. Cats: Contextually-aware thresholding for sparsity in large language models. *arXiv preprint arXiv:2404.08763*, 2024.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

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

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

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, Charles Lovering, Jason Phang, Anish Thite, Thomas Wang, et al. Eleutherai/lm-evaluation-harness: v0. 4.9. *Zenodo*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36: 62991–63010, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936, 2019.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

A Rationale for Hierarchical Budget Decomposition

The original routing problem jointly couples inter-layer budget allocation and intra-layer module selection. Our framework introduces a prompt-conditioned macro budget $\mathbf{a}(X) = [a_1(X), \ldots, a_L(X)]$ and optimizes the module-wise sparsity ratios only after these budgets have been fixed.

Let

$$E_l(\mathbf{p}_l) = \left\| \mathbf{Y}_l - \widetilde{\mathbf{Y}}_l(\mathbf{X}_l; \mathbf{p}_l) \right\|_2^2$$

denote the layer-local reconstruction surrogate. Conditioned on the macro budgets, the micro-routing problem decomposes as

$$\begin{aligned} \min_{\mathbf{p}_1, \ldots, \mathbf{p}_L} \left\{ \sum_{l=1}^{L} E_l(\mathbf{p}_l) \;\middle|\; \sum_{m \in \mathcal{M}_l} \omega_{l,m} p_{l,m} = a_l(X),\ \forall l \right\} \\ = \sum_{l=1}^{L} \min_{\mathbf{p}_l : \sum_m \omega_{l,m} p_{l,m} = a_l(X)} E_l(\mathbf{p}_l). \end{aligned} \tag{11}$$

Thus, once the macro controller resolves the inter-layer allocation, the layer-local surrogate objectives factor into independent constrained problems, each of which we approximate using a local greedy search. The factorization itself does not imply that the greedy solutions are optimal, nor does it claim global optimality under the full end-to-end model loss. Rather, it clarifies how our framework separates prompt-dependent macro budget allocation from deterministic local reconstruction.

B Algorithmic Details

In this section, we provide the complete pseudocode for the prefill-time micro routing procedure discussed in Section 3.3. Algorithm 1 details the iterative greedy search used to construct an approximate allocation of the macro-controller's assigned budget a_l^* among the internal modules of layer l and then cache the corresponding magnitude thresholds for decoding.

Algorithm 1 Prefill-Time Intra-Layer Allocation and Threshold Construction

Require: Prompt prefill input $\mathbf{X}_l$, module prefill activations $\{\mathbf{A}_{l,m}^{\text{pre}}(X)\}_m$, dense output $\mathbf{Y}_l$, macro budget a_l^*, and base footprint-weighted step size $\eta_{\text{route}} = 0.05$.
Require: Set of modules $\mathcal{M}_l = \{W_q, W_k, W_v, W_o, W_{\text{up}}, W_{\text{gate}}, W_{\text{down}}\}$.
$F_l \leftarrow \sum_{m \in \mathcal{M}_l} \text{size}(W_m)$ {Total parameter footprint of layer l}
Initialize sub-budgets $\mathbf{p}_l \leftarrow \mathbf{0}$
Initialize current layer sparsity $P_l \leftarrow 0$
while $P_l < a_l^*$ **do**
  for each module $m \in \mathcal{M}_l$ **do**
    $\delta_{l,m} \leftarrow \eta_{\text{route}} \cdot (F_l / \text{size}(W_m))$ {Calculate module-specific step size}
    $\tilde{\mathbf{p}}_l \leftarrow \mathbf{p}_l$
    $\tilde{p}_{l,m} \leftarrow \tilde{p}_{l,m} + \delta_{l,m}$
    $\tilde{\mathbf{Y}}_{(m)} \leftarrow \text{SparseLayer}_l(\mathbf{X}_l; \tilde{\mathbf{p}}_l)$ {Forward pass with trial sparsity}
    $E_m \leftarrow \|\mathbf{Y}_l - \tilde{\mathbf{Y}}_{(m)}\|_2^2$ {Evaluate MSE objective}
  end for
  $m^* \leftarrow \arg\min_m E_m$ {Greedy selection}
  $p_{l,m^*} \leftarrow p_{l,m^*} + \delta_{l,m^*}$
  $P_l \leftarrow P_l + \eta_{\text{route}}$
end while
$\hat{\mathbf{p}}_l \leftarrow \mathbf{p}_l$
for each module $m \in \mathcal{M}_l$ **do**
  $\hat{\tau}_{l,m} \leftarrow \text{Quantile}_{\hat{p}_{l,m}}\left(\left|\mathbf{A}_{l,m}^{\text{pre}}(X)\right|\right)$
end for
return $(\hat{\mathbf{p}}_l, \hat{\boldsymbol{\tau}}_l)$

Algorithm 1 is executed only during prefill. Autoregressive decoding directly consumes the cached thresholds through the fused sparse operators; neither the greedy search nor the percentile operator is invoked again.

C Pareto Frontier for All Models

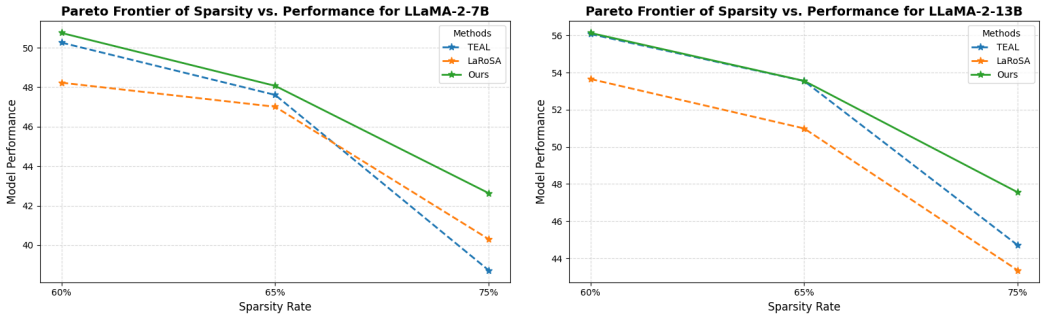

Figure 3: Pareto frontier for LLaMA-2-7B and LLaMA-2-13B.

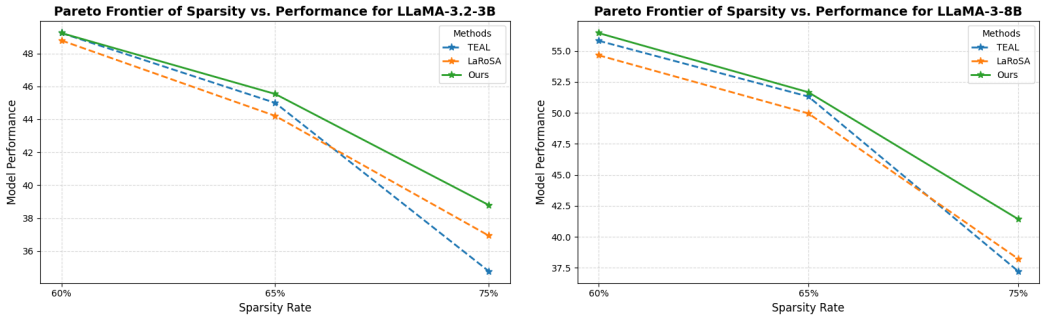

Figure 4: Pareto frontier for LLaMA-3.2-3B and LLaMA-3-8B.

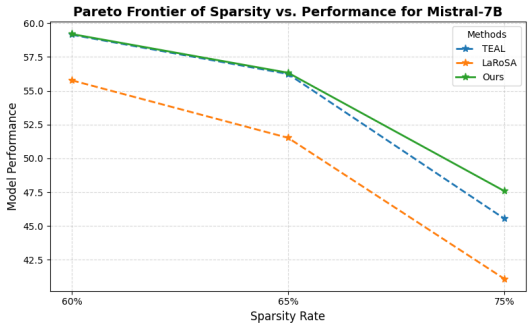

Figure 5: Pareto frontier for Mistral-7B.

D COMPARISON WITH WAS

We compare our framework against the publicly released weights of WAS (LLaMA-2-7B) at both 60% and 75% sparsity levels. As summarized in Table 4, at 60% sparsity, while WAS achieves a marginally lower PPL on the C4 dataset, our method achieves a slightly lower PPL on WikiText-2 (5.76 versus 5.81 for WAS). Under the more stringent 75% sparsity regime, ***our approach outperforms all evaluated sparse baselines on both WikiText-2 and C4***. Furthermore, comprehensive evaluations of zero-shot and few-shot downstream capabilities show that our method achieves stronger average downstream performance than WAS and the other competitors at both sparsity levels.

Table 4: Language modeling performance (Perplexity, lower is better) of LLaMA-2-7B on WikiText-2 and C4 at 60% and 75% sparsity, including a comparison with WAS. Dense performance (0% sparsity) is provided as a baseline.

Models	Sparsity	Methods	WikiText-2	C4
LLaMA-2-7B	0%	Baseline	4.87	6.67
	60%	TEAL	5.79	8.08
		LaRoSA	5.89	**7.78**
		WAS	5.81	8.06
		Ours	**5.76**	8.07
	75%	TEAL	32.35	61.43
		LaRoSA	13.43	21.37
		WAS	13.48	19.45
		Ours	**11.80**	**17.83**

Table 5: Zero-shot and few-shot downstream results of LLaMA-2-7B at 60% and 75% sparsity, including a comparison with WAS. Dense performance (0% sparsity) is provided as a baseline.

Models	Sparsity	Methods	MMLU 5-shot	C-Eval 5-shot	ARC-E 25-shot	GSM8K 5-shot	HellaSwag 10-shot	BBH 3-shot	WinoGrande 5-shot	PIQA 0-shot	BoolQ 32-shot	AVG
LLaMA-2-7B	0%	Baseline	45.68	33.28	80.68	14.03	58.92	38.83	74.66	77.80	80.40	56.03
	60%	TEAL	36.87	27.27	76.98	7.81	53.89	30.36	68.90	76.17	74.07	50.26
		LaRoSA	34.09	27.27	77.02	5.53	53.04	30.30	65.75	76.06	64.98	48.23
		WAS	36.84	28.45	76.35	7.58	54.19	30.21	67.72	76.44	74.01	50.20
		Ours	38.48	29.05	77.74	6.90	54.45	29.84	68.98	76.77	74.53	**50.75**
	75%	TEAL	25.75	27.49	53.24	0.99	30.43	26.06	52.57	71.33	60.64	38.72
		LaRoSA	26.65	24.81	59.09	1.67	35.12	27.55	53.99	72.25	61.56	40.30
		WAS	26.13	25.78	63.46	1.36	40.28	28.00	56.91	72.36	63.88	42.04
		Ours	26.34	26.30	64.64	1.36	41.69	27.60	58.96	71.55	65.23	**42.63**

E POST-HOC ANALYSIS OF TASK-DERIVED SCHEDULE TRANSFER

Complementing the independently trained prompt-free ablation in Table 3, we analyze the schedules produced by a trained prompt-conditioned policy. For each source task, we average its predicted layer allocations and reuse the resulting task-derived schedule for every example from both BBH and PIQA. These schedules are outputs of the conditional policy rather than independently optimized prompt-free baselines; this experiment therefore examines task-level schedule transfer and is not the primary causal control for prompt conditioning.

Table 6 reports this post-hoc study on LLaMA-2-13B at 75% sparsity. Native per-prompt allocation obtains the highest score on both tasks. Relative to the corresponding same-task mean schedule, the difference is 0.17 points on BBH (30.64 versus 30.47) and 0.49 points on PIQA (75.30 versus 74.81). Transferring the PIQA-derived schedule to BBH or the BBH-derived schedule to PIQA produces larger losses, suggesting that the learned allocations contain task-level preferences. Figure 6 shows where the two task means differ. Because this analysis uses one backbone, one target sparsity, two tasks, and one run, we treat it as supporting evidence rather than an estimate of a general prompt-level effect.

Protocol note. This diagnostic study was conducted in a separate evaluation run from Table 2. Its purpose is the controlled within-table comparison of allocation schedules; absolute scores are

therefore compared only within Table 6 and are not used to replace or recompute the main-table results.

Table 6: Post-hoc cross-task schedule transfer on LLaMA-2-13B at 75% sparsity. "Native" uses the allocation predicted for each prompt. Fixed schedules are task-level means derived from the trained conditional policy, not independently optimized prompt-free controls. Higher is better.

Allocation used at evaluation	BBH (3-shot)	PIQA (0-shot)	Average
Native prompt-conditioned	**30.64**	**75.30**	**52.97**
Fixed PIQA-derived schedule	30.06	<u>74.81</u>	<u>52.44</u>
Fixed BBH-derived schedule	<u>30.47</u>	73.07	51.77

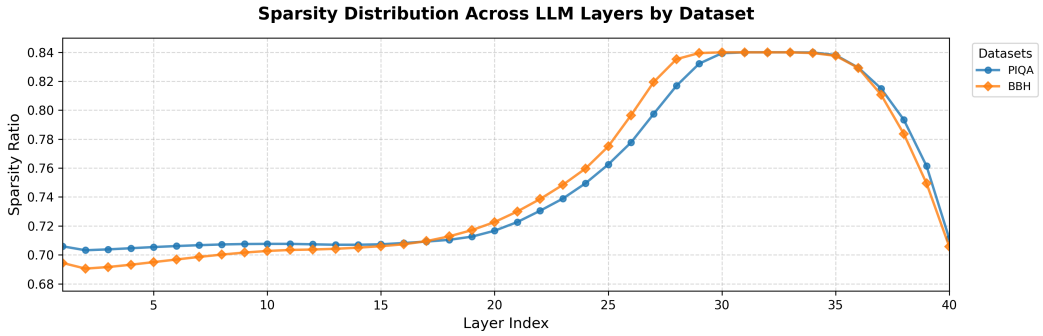

Figure 6: Task-level mean layer allocations produced for PIQA and BBH prompts in the LLaMA-2-13B study at 75% target sparsity. The curves do not show within-task prompt variation and are not independently optimized static controllers.

F Sparsity Tracing

We investigate the evolution of layer-wise sparsity in LLaMA-2-13B under global budgets of 60%, 65%, and 75%. Our results reveal that the initial 32 layers exhibit a consistent sensitivity pattern across all budgets, where sensitivity to sparsity diminishes as depth increases (manifested by the progressive rise in sparsity from Group 1 (Layers 1–8) to Group 4 (Layers 25–32)). However, the final eight layers (Group 5) demonstrate budget-dependent sensitivity: under a 60% budget, their sensitivity aligns closely with Group 2 (Layers 9–16); whereas in 65% and 75% regimes, it falls between that of Group 3 (Layers 17–24) and Group 4. These profiles show that the learned allocation structure varies with both model depth and target budget. Across the main evaluations in Tables 1 and 2, our method achieves the highest or tied-highest average downstream accuracy among the evaluated sparse methods in all 15 settings, and the lowest or tied-lowest PPL in 29 of 30 settings.

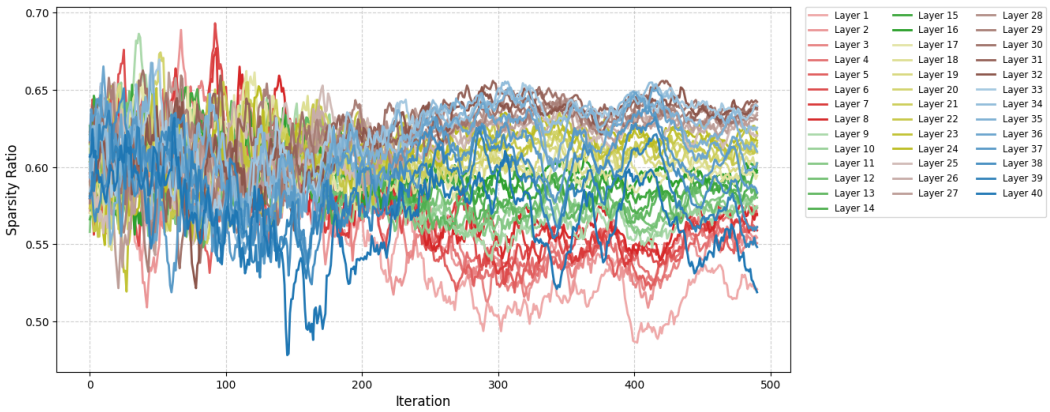

Figure 7: Layer-wise sparsity tracing for LLaMA-2-13B at 60% sparsity.

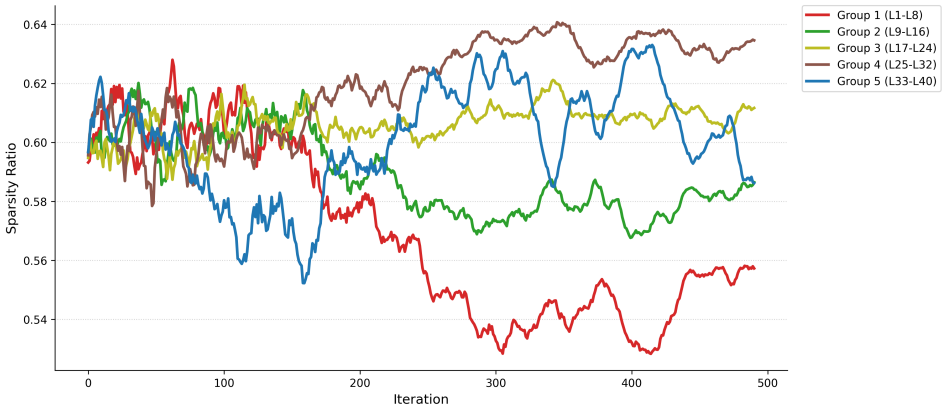

Figure 8: Group-wise sparsity tracing for LLaMA-2-13B at 60% sparsity. The 40 layers of the model are partitioned into five groups of eight layers each: G1 (L1–8), G2 (L9–16), G3 (L17–24), G4 (L25–32), and G5 (L33–40).

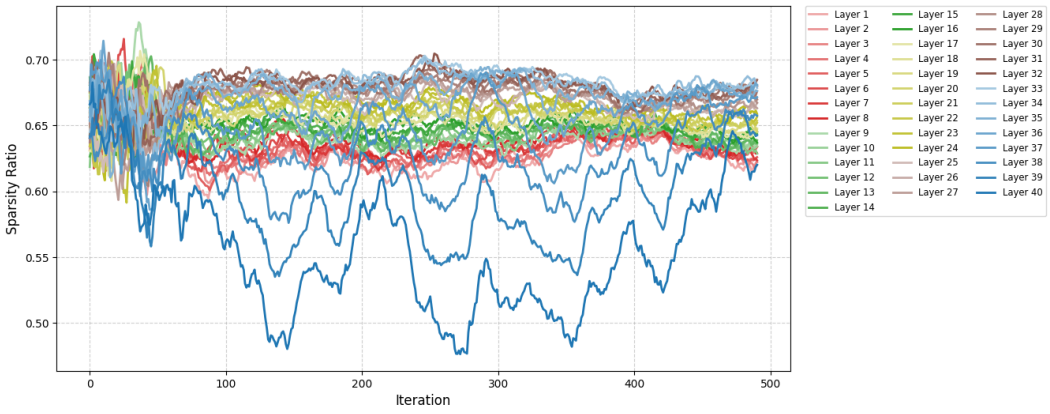

Figure 9: Layer-wise sparsity tracing for LLaMA-2-13B at 65% sparsity.

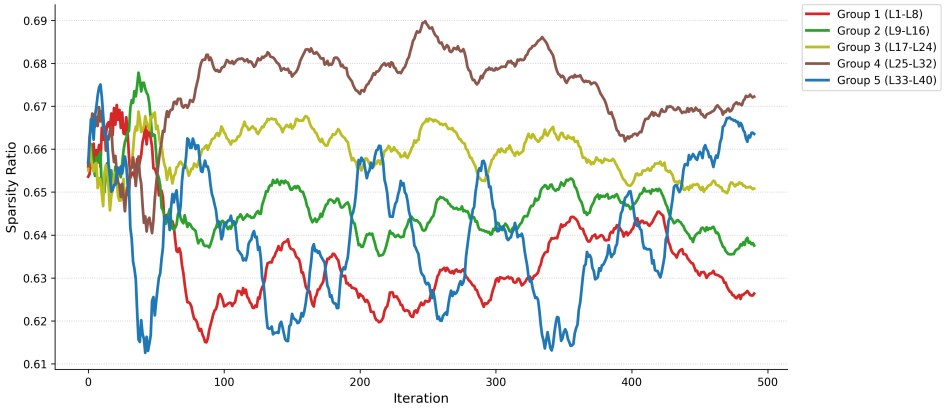

Figure 10: Group-wise sparsity tracing for LLaMA-2-13B at 65% sparsity. The 40 layers of the model are partitioned into five groups of eight layers each: G1 (L1–8), G2 (L9–16), G3 (L17–24), G4 (L25–32), and G5 (L33–40).

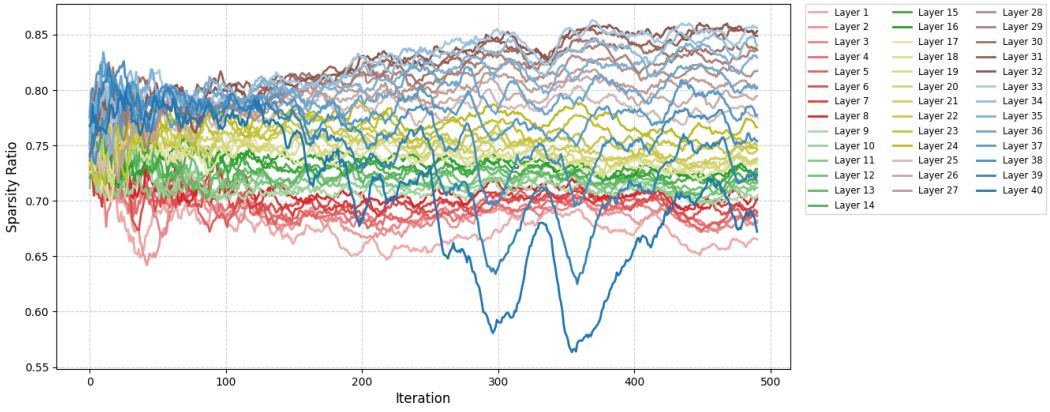

Figure 11: Layer-wise sparsity tracing for LLaMA-2-13B at 75% sparsity.

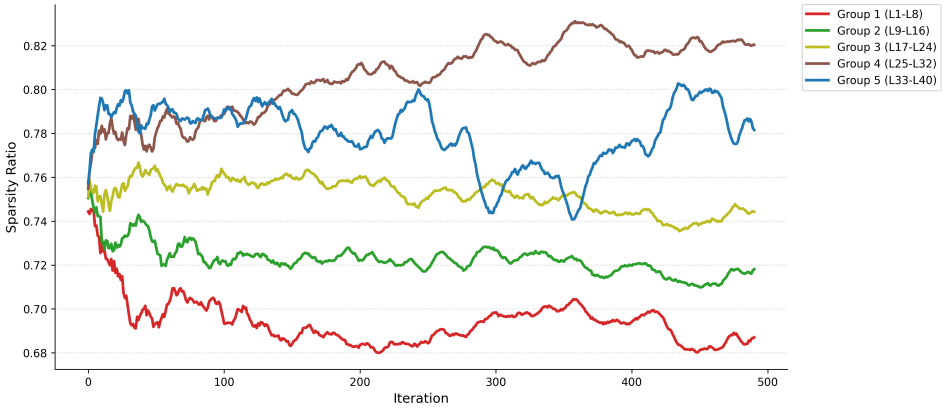

Figure 12: Group-wise sparsity tracing for LLaMA-2-13B at 75% sparsity. The 40 layers of the model are partitioned into five groups of eight layers each: G1 (L1–8), G2 (L9–16), G3 (L17–24), G4 (L25–32), and G5 (L33–40).

G Implementation Details and Policy Training Cost

Policy architecture and initialization. The actor and critic are separate three-layer MLPs with hidden dimension 256, totaling approximately 0.40M trainable parameters. The actor receives the prompt context vector and a compact, fixed-width control-state representation encoding the layer-local hidden representation, preceding sparsity decisions, remaining budget, and layer position, as defined in Section 3.1. It maps the raw mean-head output z_l to the location of the underlying normal distribution as $\mu_l = 0.95\ \mathrm{sigmoid}(z_l)$. We initialize the mean-head bias to $\mathrm{logit}(T/0.95)$, which places this pre-truncation location near target sparsity T; μ_l is not the expected value after truncation. The standard-deviation head is initialized to approximately 0.079.

Optimization. We use AdamW with learning rate 10^{-4}, 16 trajectories per update, and four PPO epochs per update. The PPO clip ratio is $\epsilon = 0.2$, the knowledge-distillation coefficient is $\alpha = 100.0$, and the sparsity-penalty coefficient is $\lambda = 100.0$. Each policy is optimized for $3{,}763$ iterations.

Under the delayed-reward formulation, every layer-wise action uses the terminal trajectory return $R_L(X)$. With critic $V_\phi(s_l)$, the advantage is

$$\hat{A}_l = R_L(X) - V_\phi(s_l). \tag{12}$$

The actor maximizes the standard clipped PPO objective

$$\mathcal{L}(\theta) = \mathbb{E}_{s_l,a_l}\left[\min\Big(\rho_l(\theta)\hat{A}_l, \mathrm{clip}(\rho_l(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_l\Big)\right], \tag{13}$$

where $\rho_l(\theta) = \pi_\theta(a_l \mid s_l)/\pi_{\theta_{\mathrm{old}}}(a_l \mid s_l)$.

Policy scope and training cost. We independently optimize one actor–critic policy for each backbone and target sparsity level. With five backbones and three sparsity targets (60%, 65%, and 75%), this yields 15 policies in total. Policies are not transferred across backbone architectures or sparsity budgets. We instead evaluate cross-dataset and cross-task generalization: policies trained exclusively on WikiText-103 are applied without further adaptation to WikiText-2, C4, and nine downstream benchmarks. No downstream labels or task-specific policy fine-tuning are used.

Each run uses 8 Ascend 910B NPUs for $3{,}763$ iterations. Depending on backbone size, a single policy requires 6.12–13.49 wall-clock hours, corresponding to 48.96–107.92 NPU-hours (Table 7). The cost is nearly invariant to the target sparsity level. Across all 15 model–sparsity configurations, the aggregate offline optimization cost is approximately $1{,}108.56$ NPU-hours. This is a one-time auxiliary-policy optimization cost and does not update the pretrained LLM.

Table 7: Offline policy-training cost per model–sparsity pair. Each run uses 8 Ascend 910B NPUs for $3{,}763$ iterations. Costs are nearly identical across the 60%, 65%, and 75% sparsity targets.

Backbone	Wall-clock (h)	NPU-hours
LLaMA-2-7B	8.68	69.44
LLaMA-2-13B	13.49	107.92
LLaMA-3-8B	9.02	72.16
LLaMA-3.2-3B	6.12	48.96
Mistral-7B	8.88	71.04

Fused sparse decoding and measurement boundary. For each prompt, the Macro Controller and Micro Router produce layer- and module-wise budgets during prefill, after which module-specific magnitude thresholds are computed and cached. During autoregressive generation, activation values, binary masks, and active neuron indices remain token dependent, whereas the prompt-conditioned budgets and thresholds remain fixed. Following WAS Wang et al. (2025), threshold comparison, mask generation, and sparse linear computation are fused into the same computation pipeline. The actor, greedy allocation, and percentile/top-k selection are absent from the decoding loop. Measurements use batch size 1 on one NVIDIA A800 and include threshold comparison, fused mask generation, and sparse computation; they exclude prefill-time policy evaluation, Micro Router search, and threshold construction. Thus, the reported numbers characterize autoregressive decoding rather than full-request latency.

H LIMITATION AND FUTURE WORK

While our approach advances backbone-frozen dynamic activation sparsity, several limitations and promising avenues remain. The current framework requires a separately optimized policy for each backbone and target sparsity level. Although this auxiliary optimization is substantially cheaper than LLM pretraining or full-parameter fine-tuning, the policies do not yet transfer across architectures or sparsity budgets. Moreover, the matched prompt-free ablation currently covers one backbone, one sparsity target, and one policy run; additional backbones, budgets, and random seeds are needed to characterize the variability and generality of the prompt-conditioning effect. The post-hoc cross-task analysis uses schedules derived from the conditional policy and should not be interpreted as an independently optimized static baseline or a within-task causal test. Developing a unified conditional policy that supports multiple backbones and target budgets is therefore an important direction for future work. In addition, extending the current prompt-level context encoder to capture fine-grained temporal dependencies in ultra-long-context or dynamic multi-turn conversational settings could further enrich the MDP state formulation. Applying this hierarchical budget allocation paradigm to Multimodal Large Language Models (MLLMs), where visual and textual tokens exhibit different redundancy patterns, also presents a compelling research direction. Finally, although our method aligns with existing sparse operators, exploring joint hardware–software co-design for prompt-customized cached-threshold execution could unlock further wall-clock acceleration.