

Rethinking Gradient Flow Through Frozen Blocks for Memory-Efficient Block-Coordinate Training

Yeachan Kim

Division of Language & AI, Hankuk University of Foreign Studies, Seoul, Republic of Korea
yeachan@hufs.ac.kr

Abstract

Block-coordinate descent (BCD) has emerged as an effective approach for memory-efficient fine-tuning of large language models. During training, BCD randomly activates a small subset of blocks, such as layers or modules, and updates only those blocks while keeping the rest frozen. Although effective, prior work has paid little attention to the frozen blocks, which must still retain forward activations to propagate gradients during the backward pass. Our analysis reveals that gradients through these frozen blocks are highly sparse at the token level: *a small subset of tokens carries nearly all of the signal required by the active blocks*. Building on this observation, we introduce frozen-block gradient approximation with sparse tokens (FROST), which approximates gradients through inactive blocks using only a sparse subset of tokens and integrates seamlessly with standard BCD optimizers. Across diverse fine-tuning scenarios, model backbones, and modalities, FROST substantially improves the memory efficiency of BCD while preserving downstream performance with negligible degradation.

1 Introduction

Large language models (LLMs) have become the dominant backbone for a wide range of natural language processing tasks, and fine-tuning remains the standard approach for adapting them to downstream domains or aligning them with human preferences. However, fine-tuning modern LLMs is often bottlenecked by GPU memory, as it requires storing not only model parameters but also gradients, optimizer states, and intermediate activations. Consequently, a wide range of methods has been proposed to reduce this memory footprint, including parameter-efficient adapters such as LoRA (Hu et al., 2021) and gradient low-rank projection (Zhao et al., 2024).

Block-coordinate descent (BCD) (Liu et al., 2025; Luo et al., 2024; Panigrahi et al., 2023) has emerged as an effective strategy for memory-efficient LLM fine-tuning. In BCD-based training, only a small subset of blocks, such as layers or modules (e.g., attention layers, MLP layers), is activated and updated at each step, while the remaining blocks are kept frozen. By rotating the active set across steps, BCD gradually covers the full parameter set while keeping optimizer states on the GPU only for the currently active blocks. This reduces optimizer memory roughly in proportion to the inactive fraction, i.e., by about $(1 - \rho)$ when ρ denotes the active-block ratio. Recent work has further improved BCD by refining active-set selection (Pan et al., 2024; Liu et al., 2025), rotation schedules (Luo et al., 2024), and combinations with parameter-efficient fine-tuning methods.

Despite its effectiveness, prior BCD methods largely overlook the cost of frozen blocks. Although these blocks are not updated, they must still retain the large forward activations so that gradients can propagate through them to reach the active blocks. As a result, activation memory remains a substantial bottleneck in BCD training. Our analysis reveals that such frozen blocks differ substantially in both memory cost and influence on the active-block update. In particular, attention modules play a critical role in propagating gradient signals, whereas inactive MLP blocks have a comparatively limited effect on the update direction and exhibit strong token-level gradient sparsity: *only a small subset of tokens carries most of the signal needed to update the active blocks*. This makes frozen MLPs a promising target for activation-memory reduction, as their backward computation can be sparsified with little effect on the descent direction.

Building on this observation, we introduce FROST, a frozen-block gradient approximation method that propagates gradients through frozen

MLP blocks using only a sparse subset of salient tokens. Rather than approximating all inactive blocks uniformly, FROST selectively targets inactive MLPs, whose contribution to active-block updates is relatively limited but whose activation memory cost remains substantial. We further find that token-level upstream gradients are highly correlated with the importance of the corresponding gradient contributions, making them an effective proxy for identifying informative tokens. FROST therefore defers token selection to the backward pass, where the upstream gradient from subsequent layers is available, and preserves the gradient signal only for the selected tokens. This allows FROST to substantially reduce the activation memory required by inactive MLPs while retaining the dominant gradient information needed for effective BCD training.

Contributions. Our contributions are summarized as follows:

- We identify frozen-block activations as a key memory bottleneck in BCD training and show that inactive MLPs are highly sparse at the token-gradient level.
- We introduce FROST, which approximates gradient propagation through frozen MLP blocks using only a subset of token representations.
- We demonstrate that FROST reduces activation memory while matching BCD accuracy, and that its adaptive token budget remains effective under tight memory constraints.

2 Related Work

Block coordinate descent (BCD) was first brought into deep neural network training as an alternative to end-to-end SGD. Specifically, Zhang and Brand (2017) reformulated Tikhonov-regularized deep neural network (DNN) training as a sequence of layer-wise subproblems solvable by BCD with provable convergence. Lau et al. (2018) introduced a proximal BCD scheme that unrolls the nested DNN structure into per-layer updates with global convergence guarantees, and Zeng et al. (2019) established general global-convergence results for BCD in common DNN training settings without requiring multi-convexity. In the LLM era, this idea has been revisited under the memory-efficiency lens: BAdam (Luo et al., 2024) partitions parameters into blocks and updates one block at a time within a BCD framework to enable full-parameter

optimization at LLM scale; LISA (Pan et al., 2024) randomly samples a few layers to activate per step based on layerwise importance, achieving accuracy comparable to or above LoRA. Beyond layer-level granularity, MISA (Liu et al., 2025) goes finer by sampling modules rather than whole layers. As empirical support for block-sparse updates, Panigrahi et al. (2023) show that fine-tuning a small task-specific subset of parameters can recover most of the gain of full fine-tuning.

All of these methods, however, focus their design on the *active* modules — which subset to update, how to rotate it across steps, and which low-rank subspace to learn in — and treat the frozen blocks as a passive, fixed cost. Consequently, they reduce only *optimizer-state* memory, while activation memory remains untouched because gradients must still flow through every inactive block to reach the active ones. To our knowledge, FROST is the first to target the *frozen*-block backward path itself as an explicit design space, making existing BCD optimizers substantially more memory-efficient.

3 Understanding Frozen Blocks in BCD

Compared to existing BCD studies that primarily focus on which blocks to activate, we instead analyze the *frozen* blocks: the activation memory they still impose and their contribution to the active-block update.¹

Frozen blocks are the remaining bottleneck.

We start by decomposing the memory footprint of BCD training in Figure 1 to identify the bottleneck that remains after optimizer-state memory is reduced through BCD. Figure 1(a) shows that BCD effectively collapses optimizer-state memory by maintaining optimizer states only for the active blocks. In contrast, activation memory remains substantial because backpropagation still needs to traverse the full computation graph, regardless of whether each block is active or inactive. Even inactive blocks, whose parameters are not updated, must transmit gradients to earlier active blocks, and thus cannot be simply removed from the backward path. As the active-parameter ratio ρ decreases, the optimizer-state footprint rapidly shrinks, but the activation footprint persists, making activation memory the dominant remaining bottleneck in BCD training.

We further decompose activation memory by ownership in Figure 1(b). The result shows that

¹Measurement setup is detailed in Appendix A.

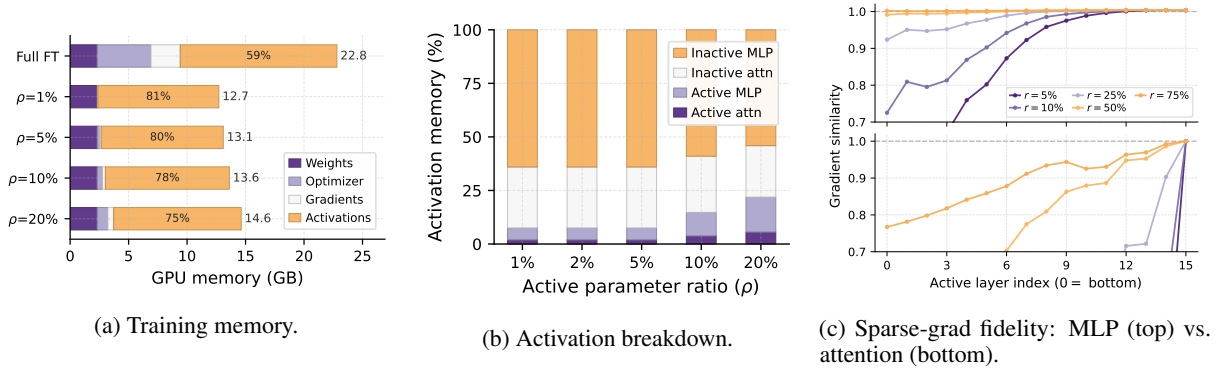


Figure 1: **(a)** BCD reduces optimizer memory, but activation memory remains a major cost. **(b)** Most activation memory comes from frozen blocks, especially inactive MLPs. **(c)** Token-sparse backward preserves frozen-MLP gradients much better than frozen-attention gradients, motivating MLP-focused compression.

most activation memory is not from the active blocks being updated, but from the frozen blocks that serve as necessary gradient pathways. Specifically, 78–93% of activation memory belongs to *frozen* blocks. This indicates that, after BCD reduces optimizer-state memory, the main remaining memory burden comes from retaining activations in frozen blocks for gradient propagation.

Frozen attention matters; frozen MLP gradients are sparse and can be well approximated. We next investigate whether the backward computation through frozen modules can be sparsified without substantially compromising the gradient received by the active module. For each experiment, we set a single active layer in the MISA (Liu et al., 2025) setup, treat the layers above it as frozen, and perform oracle token-sparse backward through the frozen modules.² Figure 1(c) reveals a clear contrast. For frozen attention (bottom), sparse backward severely distorts the active-module gradient, with cosine similarity dropping below 0.35 when keeping less than 25% of tokens and reaching only 0.89 even when keeping 75%. In contrast, frozen MLPs (top) preserve the gradient much more faithfully: even with 15 sparse MLP layers on the backward path, keeping 50% of tokens yields cosine similarity 0.95. These results indicate that frozen MLP gradients are highly concentrated at the token level, making frozen MLPs a suitable target for activation-memory reduction.

²For the oracle measurement, we first compute token-level importance using the exact non-sparse backward pass. At each frozen module, tokens are ranked by the ℓ_2 norm of their output gradient, and only the top-ranked tokens are kept for sparse backward. We then compare the gradient received by the active module under this sparse backward pass with the gradient from the original non-sparse backward pass.

4 FROST: Frozen-Block Gradient Approximation with Sparse Tokens

In this section, we introduce FROST, which reduces the activation memory of inactive blocks by propagating gradients through only a small subset of token positions. We first define BCD-based fine-tuning and the gradient path through inactive sub-blocks (§4.1). We then present FROST’s selection rule, which deterministically selects salient tokens based on the upstream-gradient norm (§4.2). Finally, we introduce an adaptive token-budget scheme that allocates more tokens to layers whose gradient corrections are less concentrated (§4.3).

4.1 Preliminaries and Problem Formulation

Block-coordinate descent. Consider a Transformer with parameters $\theta = (\theta_1, \dots, \theta_M)$ partitioned at the granularity of decoder layers. At each step s , a BCD optimizer activates a subset $\mathcal{A}_s \subseteq \{1, \dots, M\}$ and updates only $\theta_{\mathcal{A}_s}$:

$$\theta_{\mathcal{A}_s}^{(s+1)} = \theta_{\mathcal{A}_s}^{(s)} - \eta \nabla_{\theta_{\mathcal{A}_s}} \mathcal{L}(\theta^{(s)}), \quad (1)$$

with $\theta_{\mathcal{A}_s^c}^{(s+1)} = \theta_{\mathcal{A}_s^c}^{(s)}$, where $\mathcal{A}_s^c := \{1, \dots, M\} \setminus \mathcal{A}_s$ denotes the *inactive* (frozen) modules at step s . Cycling $\mathcal{A}_s$ across steps recovers full-parameter coverage while keeping optimizer state only for the active block. Existing BCD variants differ in how they pick $\mathcal{A}_s$ — LISA (Pan et al., 2024) samples layers uniformly per step, MISA (Liu et al., 2025) picks modules by importance, BAdam (Luo et al., 2024) cycles a fixed schedule — but they all share the same gradient-propagation requirement on $\mathcal{A}_s^c$. FROST is agnostic to this choice: it acts only on the backward through $\mathcal{A}_s^c$ and is therefore a drop-in addition to any of these BCD rules.

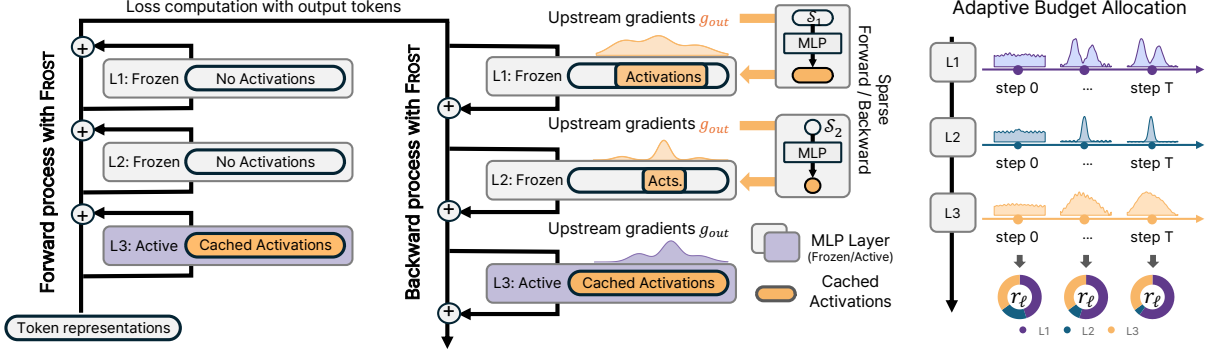


Figure 2: Overview of FROST. Frozen MLPs discard forward activations and, during the backward pass, recompute exact Jacobian-vector products only for tokens selected by $\|g_{out}\|_2$; the remaining tokens use the residual identity path. A rank-adaptive allocator redistributes the global token budget across layers.

Gradient approximation on frozen blocks.

Among the modules in $\mathcal{A}_g^c$, frozen MLP blocks dominate both the activation memory and the per-token backward cost (§3). Consider an inactive MLP block on the residual edge (we omit Layer-Norm for brevity):

$$h_{out,t} = h_{in,t} + \text{MLP}(h_{in,t}), \quad (2)$$

and write $g_{out,t} = \partial\mathcal{L}/\partial h_{out,t}$ for the upstream gradient at token t . By the chain rule, the exact MLP backward contributes a per-token Jacobian correction

$$u_t := g_{out,t} J_{\text{MLP}}(h_{in,t}), \quad (3)$$

which the inactive MLP must compute for *every* position t — the cost §3 attributes to frozen MLPs. Given a token subset $\mathcal{S} \subseteq \{1, \dots, T\}$ of size $k = \lfloor r \cdot T \rfloor$ with keep ratio $r \in (0, 1]$, FROST’s goal is to replace this exact backward with the token-sparsified rule

$$\tilde{g}_{in,t} = \begin{cases} g_{out,t} + u_t & \text{if } t \in \mathcal{S}, \\ g_{out,t} & \text{otherwise.} \end{cases} \quad (4)$$

Tokens in $\mathcal{S}$ retain the exact MLP backward, while tokens outside drop the Jacobian correction u_t and keep only the residual-skip gradient, so the backward graph remains intact at every position. Our objective, then, is to choose $\mathcal{S}$ so that (4) most faithfully approximates the exact MLP backward under the keep-ratio budget $k = \lfloor r \cdot T \rfloor$ — i.e., we seek the size- k token subset that minimizes the approximation error.

4.2 Gradient-Informed Token Selection

Ideally, the selector should keep tokens with large MLP gradient corrections. However, these correc-

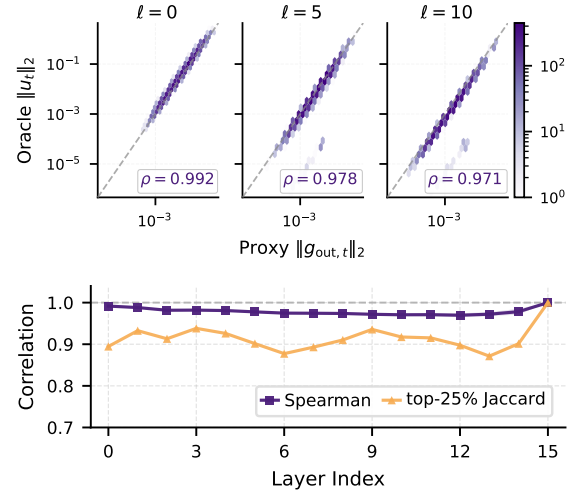


Figure 3: Correlation between the backward-time proxy $\|g_{out,t}\|_2$ and the oracle token importance $\|u_t\|_2$. The upstream-gradient norm provides a reliable proxy for identifying informative tokens during the backward.

tions cannot be reliably identified during the forward pass, since they depend on the upstream gradient produced by subsequent layers. We observe that the norm of each token’s upstream gradient is highly correlated with the importance of its gradient correction (Figure 3), making it an effective proxy for identifying informative tokens. FROST therefore defers token selection to the backward pass.

Concretely, inactive MLPs store no intermediate activations during the forward pass and simply pass their outputs to the next layer. Once the backward pass reaches an inactive MLP, FROST scores each token by the norm of its upstream output gradient and deterministically selects the top- k tokens as

the kept subset $\mathcal{S}$:

$$\mathcal{S} = \text{Top-}k\{\|g_{\text{out},t}\|_2 \mid t = 1, \dots, T\}. \quad (5)$$

FROST then recomputes MLP activations only for the selected tokens and computes their input-gradient correction. This eliminates the need to store inactive-MLP activations during the forward pass while still enabling gradient-informed token selection. Although recomputation introduces extra forward computation, it is applied only to a sparse subset of tokens. Thus, the overhead remains limited at practical keep ratios; for small keep ratios, the partial-forward and sparse-backward path can even be cheaper than a standard full inactive-MLP backward pass (detailed in §5.9).

4.3 Rank-Adaptive Gradient Approximation

A uniform keep ratio r assumes that every frozen MLP has the same token-level sparsity, but the correction energy is much more diffuse in some layers than others (as can be seen in Figure 1). We therefore keep the same global token budget but allocate more tokens to layers whose correction energy is less concentrated. For each inactive layer ℓ , we measure concentration with the Shannon effective rank of the normalized correction energy:

$$p_{\ell,t} = \frac{\|u_{\ell,t}\|^2}{\sum_{t'} \|u_{\ell,t'}\|^2}, \quad (6)$$

$$\text{EffRank}_\ell = \exp\left(-\sum_{t=1}^T p_{\ell,t} \log p_{\ell,t}\right). \quad (7)$$

Larger EffRank_ℓ indicates a more diffuse correction and therefore a larger token budget.

Based on this rank, we maintain an exponential moving average $\hat{e}_\ell$ of EffRank_ℓ and periodically allocate budget over the current inactive set $\mathcal{I}_s = \{\ell : \theta^{(\ell)} \in \theta_{\mathcal{A}_s^c}\}$:

$$\tilde{r}_\ell = |\mathcal{I}_s| \frac{\hat{e}_\ell}{\sum_{\ell' \in \mathcal{I}_s} \hat{e}_{\ell'}}. \quad (8)$$

Since the layer-wise shares are updated from gradient statistics observed during training, the token budget can dynamically adapt to the current training state. The shares sum to $|\mathcal{I}_s|$, so the average keep ratio remains r , while the BCD active-set rule and optimizer are otherwise unchanged.

5 Experiments

5.1 Setup

Scenarios. We evaluate FROST across four fine-tuning scenarios that span common downstream use cases: (i) *commonsense reasoning* — PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), WinoGrande (WNGD) (Sakaguchi et al., 2021), HellaSwag (HS) (Zellers et al., 2019), OpenBookQA (OBQA) (Mihaylov et al., 2018), StrategyQA (SQA) (Geva et al., 2021), and CommonsenseQA (CSQA) (Talmor et al., 2019); (ii) *mathematical reasoning* — GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and AQuA (Ling et al., 2017), formulated as chain-of-thought supervision; (iii) *instruction tuning* on broad-capability instruction-following data, evaluated on MMLU and BBH; and (iv) *multimodality*, where we transfer the same inactive-MLP backward to non-language Transformer encoders.

Backbones. Each scenario uses a backbone matched to its supervision style: Llama-3.2-1B-Instruct (Grattafiori et al., 2024) and Qwen2.5-3B-Instruct (Yang et al., 2024) for commonsense reasoning, Qwen2.5-Math-1.5B (Yang et al., 2024) for mathematical reasoning, and Llama-3.2-3B (Grattafiori et al., 2024) for instruction tuning. Training uses bf16 mixed precision with a bf16-state AdamW optimizer; learning rate, batch size, and epoch count are tuned per task and held fixed across all baselines on each task.

Baselines. We compare FROST against four families of methods: (i) FULL FT — all parameters trainable; (ii) LORA (Hu et al., 2021) — rank-8 adapters; (iii) LISA (Pan et al., 2024) — layerwise importance sampling, a BCD variant that activates a random subset of layers per step; (iv) MISA (Liu et al., 2025) — module-level BCD at $\rho=0.05$ active-parameter ratio with the default sample_last=2 schedule. Because FROST targets the inactive-block backward of any BCD method, we evaluate it as a drop-in addition on top of both LISA and MISA, denoted LISA + FROST and MISA + FROST. For FROST, we use a keep ratio of $r = 0.5$ in the main experiments, which works reliably across most tasks. Validation results with different keep ratios are provided in Figure 7.

Metrics. For each method we report task accuracy averaged over 3 seeds and peak GPU memory

Table 1: Commonsense-reasoning accuracy (%) and peak memory (GB). The best and second-best results are highlighted in **boldface** and underlined, respectively.

Method	PIQA	ARC-e	ARC-c	WNGD	HS	OBQA	SQA	CSQA	Avg	Mem
<i>Backbone: Llama-3.2-1B-Instruct</i>										
Full FT	80.4	81.3	61.7	72.9	86.1	75.1	63.2	72.4	<u>74.1</u>	22.4
LoRA (Hu et al., 2021)	78.4	75.7	58.8	70.7	83.3	72.8	60.9	70.1	71.3	14.8
LISA (Pan et al., 2024)	78.1	79.3	50.4	67.0	82.8	72.0	62.7	69.1	70.2	17.4
+ FROST (ours)	77.7	79.4	50.5	69.9	83.1	71.9	62.7	69.4	70.6	<u>12.5</u>
MISA (Liu et al., 2025)	80.2	80.9	<u>61.0</u>	72.4	85.3	<u>75.2</u>	<u>64.3</u>	73.4	<u>74.1</u>	<u>13.7</u>
+ FROST (ours)	80.2	<u>81.1</u>	61.2	<u>72.7</u>	<u>85.2</u>	75.4	64.8	<u>73.1</u>	74.2	9.5
<i>Backbone: Qwen2.5-3B-Instruct</i>										
LoRA (Hu et al., 2021)	86.8	93.1	81.9	84.5	<u>93.0</u>	87.4	68.6	83.4	84.8	<u>17.9</u>
LISA (Pan et al., 2024)	85.0	93.9	82.5	76.4	<u>92.0</u>	85.2	71.3	82.6	83.6	21.9
+ FROST (ours)	85.5	93.7	82.8	76.6	91.0	85.6	72.0	81.7	83.6	19.9
MISA (Liu et al., 2025)	87.7	<u>94.7</u>	83.1	83.4	92.9	88.4	74.7	84.4	86.2	18.9
+ FROST (ours)	<u>87.1</u>	95.0	83.2	83.2	93.3	<u>88.2</u>	<u>74.3</u>	<u>84.2</u>	<u>86.1</u>	13.5

(torch.cuda.max_memory_allocated, GB) during a steady-state training step.

5.2 Main results: commonsense reasoning

Table 1 shows a consistent accuracy–memory trade-off across the commonsense benchmarks. Although BCD performs full-parameter fine-tuning, it is already highly memory-efficient by updating only a subset of blocks at each step. In our results, BCD baselines often outperform parameter-efficient tuning methods such as LoRA while using less peak memory. Adding FROST further improves this trade-off: regardless of the underlying BCD strategy, FROST consistently reduces peak memory while preserving average accuracy. Across the completed settings, FROST lowers peak memory by roughly 25–34% relative to the corresponding BCD baseline, with little to no accuracy degradation. This trend holds for both importance-based (MISA) and sampling-based BCD (LISA) variants, suggesting that FROST improves the inactive-block backward path rather than depending on a particular active-set rule. This supports the main hypothesis that frozen MLP activations are a persistent and reusable compression target in BCD-based training.

5.3 Main results: mathematical reasoning

For this experiment, we fine-tune the model on Math10K (Hu et al., 2023) and evaluate it on held-out math-reasoning benchmarks. Table 2 shows that FROST preserves the accuracy profile of its BCD baseline while reducing peak memory by about 20%. It also remains more memory-efficient than LoRA while achieving comparable or slightly

Table 2: Math-reasoning accuracy and peak memory on Qwen2.5-Math-1.5B fine-tuned on Math10K.

Method	GSM8K	SVAMP	AQuA	Avg	Mem
Pre-trained	52.2	65.2	22.2	46.5	—
Full FT	<u>76.6</u>	74.4	41.3	64.1	23.9
LoRA (Hu et al., 2021)	76.7	73.2	40.1	63.3	13.0
MISA (Liu et al., 2025)	76.5	74.6	42.1	64.4	<u>11.0</u>
+ FROST (ours)	76.4	<u>74.5</u>	<u>41.9</u>	<u>64.3</u>	9.0

Table 3: Instruction-tuning accuracy and peak memory.

Method	MMLU	BBH	Avg	Mem
LoRA (Hu et al., 2021)	54.7	46.2	50.5	23.2
LISA (Pan et al., 2024)	55.7	47.4	51.6	28.0
+ FROST (ours)	55.2	47.2	51.2	<u>22.3</u>
MISA (Liu et al., 2025)	<u>55.6</u>	<u>47.3</u>	51.5	24.1
+ FROST (ours)	55.5	47.6	51.6	16.5

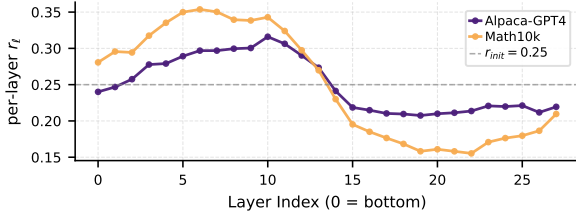
better average accuracy. These results suggest that sparse inactive-MLP backward is not limited to classification-style supervision and remains effective for chain-of-thought fine-tuning.

5.4 Main results: instruction tuning

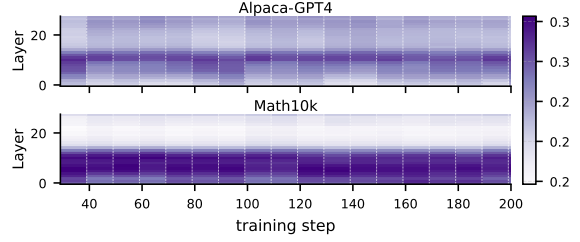
Table 3 shows the same pattern for instruction fine-tuning. On both benchmarks, adding FROST to BCD improves memory efficiency without compromising accuracy. These results further support that the token-level approximation for frozen MLP blocks is broadly applicable across diverse fine-tuning scenarios.

5.5 Per-layer budget distribution

We inspect the per-layer keep ratios r_ℓ chosen by the adaptive allocator during training. As shown in Figure 4, the allocation stabilizes almost immediately: after the first quartile of training, the



(a) Per-layer budget.



(b) Budget allocation during training.

Figure 4: Per-layer keep ratio r_ℓ for Alpaca-GPT4 and Math10K.

mean temporal standard deviation per layer is only 0.009, with maximum drift 0.06. The steady-state profile is also consistent across Alpaca-GPT4 and Math10K: both allocate more tokens to lower layers and fewer to upper layers, with a transition near mid-depth, while Math10K shows a larger amplitude than Alpaca-GPT4. This suggests that the adaptive budget mainly captures a stable depth-dependent concentration pattern, so it generalizes across tasks without per-task tuning.

5.6 Cross-modal generalization of FROST

FROST is not specific to decoder-only LLMs; it applies equally to Transformer models for other modalities. We thus evaluate on ViT-Base/16 (Dosovitskiy et al., 2021) for image classification (Pets-37, Flowers-102) and AST-Base (Gong et al., 2021) for audio classification (ESC-50, SC-V2); full experimental setup is deferred to Appendix B. Table 4 shows that FROST transfers cleanly to both modalities: applied to MISA it preserves accuracy for all tasks while reducing peak memory by 24.6–25.0% relative to MISA. These results suggest that FROST exploits a modality-agnostic property of Transformer MLP blocks rather than an LLM-specific artifact.

5.7 Ablation study

Table 5 ablates the two key design choices of FROST—gradient-informed token selection and adaptive token budgeting—on two representative datasets. We replace the backward-time selection score with three forward-time alternatives (Uniform, Norm, and Attention), and also evaluate a variant without the adaptive budget. FROST outperforms all ablations on both tasks, showing that both components are important. The forward-time alternatives perform worse because they rank tokens using signals only loosely aligned with the MLP gradient correction $\|u_t\|_2$. Removing the adaptive budget also degrades performance, indicating that

Table 4: Cross-modal accuracy and peak memory.

Method	Pets-37		Flowers-102	
	Acc	Mem	Acc	Mem
LoRA (Hu et al., 2021)	92.4	20.8	93.5	20.8
MISA (Liu et al., 2025)	92.3	19.9	93.2	19.9
+ FROST (ours)	92.2	15.0	93.2	15.0

(a) Vision (ViT-Base/16)

Method	ESC-50		SC-V2	
	Acc	Mem	Acc	Mem
LoRA (Hu et al., 2021)	93.3	16.0	86.4	16.0
MISA (Liu et al., 2025)	95.3	15.8	86.6	16.0
+ FROST (ours)	95.5	11.9	86.3	12.0

(b) Audio (AST-Base)

Table 5: Ablation study on FROST with various selection rules and adaptive budget allocation.

Variant	PIQA	GSM8K
FROST (ours)	80.2	76.4
w/ Uniform Sampling (fwd)	77.0	70.5
w/ Norm-based Sampling (fwd)	78.5	73.0
w/ Attention-based Sampling (fwd)	78.0	72.0
w/o adaptive budget	78.7	74.5

dynamically allocating token budgets across layers is important for preserving accuracy under sparse backward propagation.

5.8 Training throughput

Figure 5 compares the scalability of LoRA, MISA, and MISA + FROST on a fixed 24 GiB GPU. LoRA and MISA reach similar OOM limits because both still retain the full activation stack for backpropagation. In contrast, MISA + FROST reduces the inactive-MLP activation burden and substantially extends both scaling axes, increasing the maximum batch size from 31 to 64 and the maximum sequence length from 1,280 to 2,688. Importantly, in the practical regime where FROST works well, such as $r = 0.5$, this memory gain comes with

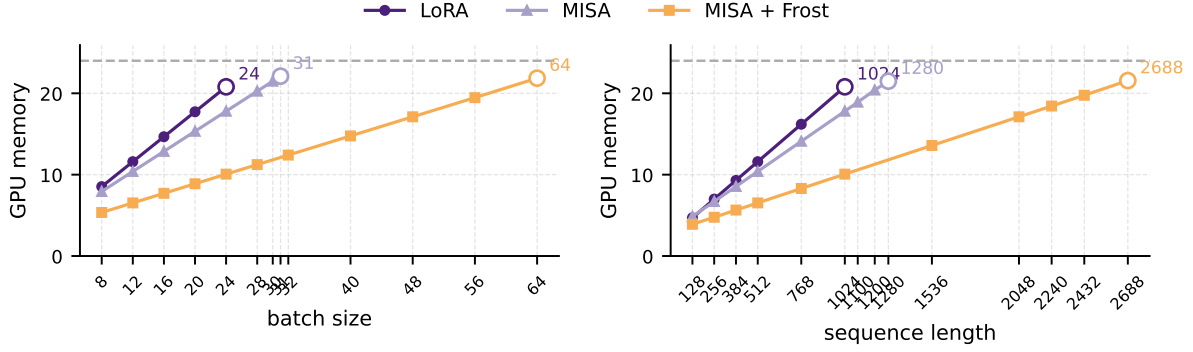


Figure 5: Peak GPU memory as batch size (left) and sequence length (right) scale on a fixed 24 GiB GPU; each line ends at the last setting before OOM.

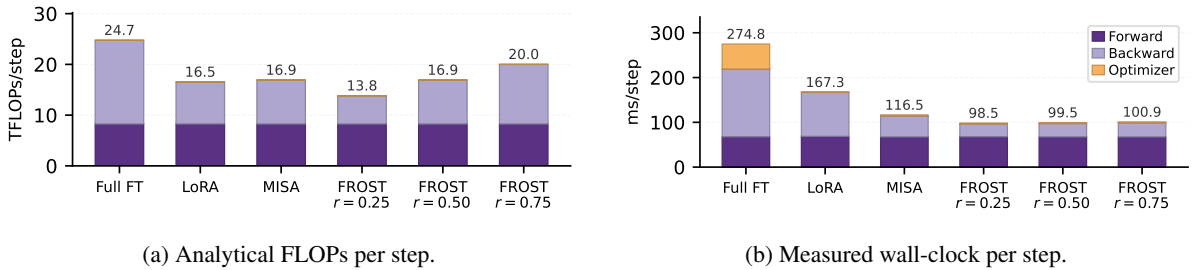


Figure 6: Per-step cost of Full FT, LoRA, MISA, and FROST at three keep ratios, stacked by forward / backward / optimizer phase.

almost no throughput overhead compared with MISA. Thus, FROST expands the feasible batch and context length without sacrificing the efficiency of the underlying BCD optimizer.

5.9 Computational overheads of FROST

Although FROST avoids storing inactive-MLP activations during the forward pass, it still recomputes activations for the selected token subset during the backward pass. This subset recomputation introduces additional computation, so we analyze whether the memory savings come with practical compute overhead. To this end, Figure 6 measures both analytical FLOPs and wall-clock time under different keep ratios r . The results show that FROST adds little overhead in the practical regime. Relative to MISA, the analytical FLOPs are negative or near-zero at smaller keep ratios: -18% at $r = 0.25$ and 0% at $r = 0.5$, increasing to $+18\%$ only at the more aggressive $r = 0.75$ setting. Wall-clock measurements are even more favorable: FROST is 15–19 ms faster per step than MISA across all keep ratios, as the sparse inactive-MLP path avoids autograd-graph construction for frozen MLPs. Overall, the activation-memory re-

duction of FROST comes with negligible practical compute overhead, especially at the default $r = 0.5$ setting used in our main experiments.

6 Conclusion

In this paper, we have identified frozen-block activations as a major remaining bottleneck in BCD fine-tuning and shown that inactive MLPs provide an effective compression target due to their token-level gradient sparsity. Based on this observation, we have introduced FROST, a gradient-informed sparse backward method that preserves the dominant gradient signal through selected tokens, together with an adaptive layer-wise token budget for handling less concentrated gradients. Across language, vision, and audio Transformers, FROST has substantially reduced peak memory while preserving the accuracy of BCD, highlighting inactive-MLP backward as a broadly useful design space for memory-efficient training.

Limitations

Inactive attention is left untouched. FROST focuses on inactive MLPs because our analysis shows that their gradients are highly sparse at the token

level and can be approximated well with a small subset of tokens. In contrast, inactive attention appears much more sensitive to token sparsification: reducing its backward computation substantially distorts the gradient received by active blocks. For this reason, we leave inactive attention unchanged in the current method. Developing sparsification rules for inactive attention, while accounting for the coupling among tokens induced by self-attention, is an important direction for future work.

Scaling to larger models. Our experiments cover multiple fine-tuning scenarios, model backbones, and modalities, but they do not include models beyond the 10B-parameter scale. Since larger models may exhibit different activation-memory profiles, gradient concentration patterns, or system-level overheads, evaluating FROST on 10B+ models remains an important next step. Nevertheless, because FROST targets inactive-block activation memory, which grows with model depth and sequence length, we expect its benefits to remain relevant at larger scales.

Acknowledgment

This work was supported by Hankuk University of Foreign Studies Research Fund (of 2026). This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00415812, RS-2026-25606011).

References

- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*.
- 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. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Yuan Gong, Yu-An Chung, and James Glass. 2021. AST: Audio spectrogram transformer. In *Inter-speech*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.
- Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *EMNLP*.
- Tim Tsz-Kit Lau, Jinshan Zeng, Baoyuan Wu, and Yuan Yao. 2018. A proximal block coordinate descent algorithm for deep neural network training. *arXiv preprint arXiv:1803.09082*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.
- Yuxi Liu, Renjia Deng, Yutong He, Tao Yao, Kun Yuan, and 1 others. 2025. Misa: Memory-efficient llms optimization with module-wise importance sampling. In *NeurIPS*.
- Qijun Luo, Hengxu Yu, and Xiao Li. 2024. Badam: A memory efficient full parameter optimization method for large language models. In *NeurIPS*.
- 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. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391.
- Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layer-wise importance sampling for memory-efficient large language model fine-tuning. In *NeurIPS*.
- Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. In *ICML*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *NAACL*.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL*.

An Yang and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Jinshan Zeng, Tim Tsz-Kit Lau, Shaobo Lin, and Yuan Yao. 2019. Global convergence of block coordinate descent in deep learning. In *ICML*.

Ziming Zhang and Matthew Brand. 2017. Convergent block coordinate descent for training Tikhonov regularized deep neural networks. In *NeurIPS*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *ICML*.

A Measurement Setup for §3

All measurements reported in §3 use **Llama-3.2-1B-Instruct** on **PIQA**, configured as a sequence-classification fine-tuning task with batch size $B=12$, sequence length $T=512$, and bf16 mixed-precision training; numbers are averaged over 6 randomly drawn training batches and across all 16 decoder layers.

B Cross-Modal Experimental Setup

This section details the experimental configuration for the cross-modal results in §5.6 (Table 4).

Backbones and adaptation. Vision experiments use ViT-Base/16 (Dosovitskiy et al., 2021); audio experiments use AST-Base (Gong et al., 2021). For both backbones we apply rank-8 LoRA on the attention $\{q, v\}$ projections with the classifier head left trainable and MLP weights frozen, matching the inactive-MLP setting that FROST targets.

Datasets and token lengths. Vision experiments use Oxford-IIIT Pet (Pets-37) and Flowers-102 with $B=256$ and $T=197$ patches. Audio experiments use ESC-50 and SpeechCommands V2 (SC-V2) with $B=32$ and $T=1,024$ frames. Batch sizes are chosen as the largest values that fit a single 24 GiB GPU under vanilla LoRA, so the peak-memory comparison is reported in a high-utilization setting rather than a slack regime.

C Effect of Keep Ratio r on MISA + FROST

The keep ratio r controls the per-layer token budget $K_\ell = \lfloor r_\ell \cdot r \cdot T \rfloor$ (§4.3): smaller r keeps fewer tokens per inactive MLP and yields a more aggressive activation sparsification. Figure 7 sweeps r on two tasks of differing supervision and backbone — Llama-3.2-1B-Instruct / PIQA ($r \in \{0.10, 0.15, 0.25, 0.375, 0.50, 0.75\}$, sequence classification, 3 epochs, $B=12$, $T=512$) and Qwen2.5-Math-1.5B / Math10K $\rightarrow$ GSM8K ($r \in \{0.10, 0.25, 0.50, 0.75\}$, causal-LM with chain-of-thought, 1,500 optimizer steps, $B=2$, gradient accumulation 4) — otherwise matching the main-text configurations (adaptive per-layer budget with $r_{\min}=0.1r$, $r_{\max}=2r$, EMA decay 0.9, warmup 100 steps).

Both curves share the same plateau shape, despite the different backbones, task formulations, and reference accuracies. **(i)** Above the transition point $r=0.25$, accuracy increases monotonically and saturates at $r \in [0.5, 0.75]$, matching the vanilla-MISA reference on both tasks (PIQA: 80.2% at $r=0.5$ vs. 80.2%; GSM8K: 76.0% at $r=0.5$ vs. 74.6%). **(ii)** Below $r=0.25$ accuracy collapses sharply: at $r=0.10$ the keep ratio is squeezed against the lower clipping bound $r_{\min}=0.1r$, so the adaptive allocator can no longer redistribute meaningfully and many shallow layers fall below the minimum tokens needed to estimate their correction; the resulting gradient signal is too noisy and training stalls (72.7% on PIQA, 56.7% on GSM8K — −7 and −18 points below the respective MISA references). **(iii)** The accuracy plateau in $r \in [0.375, 0.75]$ is therefore the operating regime where FROST delivers the largest activation-memory savings per accuracy point lost, supporting the $r=0.5$ default used throughout the main paper as a backbone- and task-independent choice.

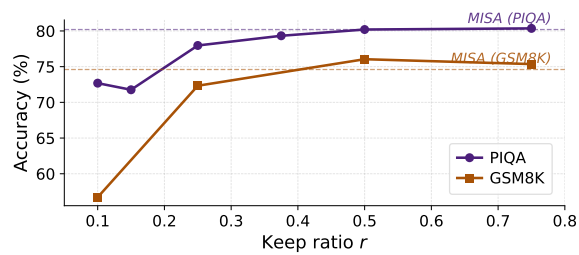


Figure 7: MISA + FROST accuracy vs. keep ratio r for two backbones / tasks — PIQA (Llama-3.2-1B, sequence classification) and GSM8K (Qwen2.5-Math-1.5B fine-tuned on Math10K, causal-LM CoT). Dotted horizontal lines mark each task’s vanilla-MISA reference. Both curves exhibit the same plateau shape above $r=0.25$ and collapse at $r=0.10$.