Dominant-Layer ZO: A Single Layer Dominates Zeroth-Order Fine-Tuning of LLMs

Wanhao Yu[1] **Ziyan Wang**[1] **Zheng Wang**[2] **Abeer Matar Almalky**[3]
Yihang Zuo[4] **Shuteng Niu**[5] **Sen Lin**[2] **Adnan Siraj Rakin**[3]
Deliang Fan[4] **Li Yang**[1†]

[1]University of North Carolina at Charlotte [2]University of Houston
[3]State University of New York at Binghamton [4]Arizona State University
[5]Texas Christian University

Abstract

Zeroth-order (ZO) optimization enables memory-efficient fine-tuning of large language models (LLMs) using only forward passes, but it remains unclear how useful adaptation is distributed across layers. In this work, we reveal a surprising phenomenon: ZO fine-tuning is sharply dominated by a single decoding layer. Across multiple LLM families and downstream tasks, fine-tuning this dominant layer alone consistently matches or even exceeds full-model ZO fine-tuning. We further show that the dominant layer is task-agnostic but model-specific, and can be identified before training through a simple inference-only analysis of activation outliers. Specifically, the dominant layer consistently aligns with the first activation-outlier layer in the pre-trained model. To explain this phenomenon, we analyze how perturbation effects propagate under ZO optimization. We find that the dominant layer combines two key properties: high perturbation sensitivity and early placement in the residual stream, allowing perturbation-induced effects to propagate and accumulate through remaining subsequent decoding layers. As a result, this layer produces stronger optimization signals under forward-only updates. Extensive experiments on LLaMA2-7B and Qwen3-8B across nine benchmarks show that dominant-layer ZO fine-tuning improves average performance over full-model MeZO and LoRA-based ZO fine-tuning while achieving up to 4.52× training speedup. Code is available at `https://github.com/uncc-efficient-ai/dominant-layer-zo`.

1 Introduction

Zeroth-order (ZO) optimization has recently emerged as a promising approach for memory-efficient fine-tuning of large language models (LLMs) [43]. Instead of computing first-order (FO) gradients through backpropagation, ZO methods estimate update directions using only forward evaluations, typically by measuring loss differences under random parameter perturbations [32]. Building on this idea, MeZO shows that pre-trained LLMs can be fine-tuned with near inference-level memory [24]. Subsequent studies further improve convergence and accuracy by reducing the variance of ZO gradient estimates through sparse parameter perturbation [23, 15], low-rank or structured perturbation spaces [6, 22], and more stable or faster optimizer designs [5, 9]. Despite these advances, existing methods largely treat ZO fine-tuning as a full-model process, without explaining how useful adaptation differs across layers. This leaves a fundamental question unanswered: *under forward-only updates, where does useful optimization actually occur inside LLM architectures?*

†Corresponding author.

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

In this work, we reveal an unexpected phenomenon: **useful ZO adaptation is not spread broadly across layers, but is dominated by a single layer.**

To study this phenomenon, we first conduct a systematic layer-wise analysis across multiple LLMs and downstream tasks, where we fine-tune one layer at a time under identical ZO updates while freezing all other layers. The results show a highly uneven layer-wise pattern: most layers provide little or no improvement over the no fine-tuning baseline, while a single layer consistently achieves performance comparable to, or even exceeding, full-model ZO fine-tuning. We refer to this layer as ***dominant layer***. Moreover, the dominant layer is *task-agnostic but model-specific*: for a given LLM, the same layer consistently dominates across the evaluated tasks, while different model families may have different dominant-layer indices. In contrast, under matched first-order gradient fine-tuning, improvements are more evenly spread across layers, and no single layer consistently dominates. This contrast shows that the dominant-layer phenomenon is unique for ZO optimization, which follows a different layer-wise adaptation pattern from first-order (FO) fine-tuning.

We further study how to efficiently identify this dominant layer without expensive layer-wise ZO fine-tuning. Inspired by the known *activation outlier* phenomenon in LLMs [11, 37], where a small number of activations exhibit extremely large magnitudes at specific dimension indices across layers in an input-independent manner [33, 2], we find that the dominant layer aligns with the first layer where activation outliers emerge. Based on this observation, we design a simple inference-only selection method: given a small calibration set, we run the pre-trained LLM forward, measure layer-wise activation statistics, and select the first layer that shows a clear outlier pattern. This method avoids exhaustive layer-wise ZO fine-tuning and identifies the dominant layer before training begins.

Finally, we explain why this dominant layer emerges under ZO fine-tuning. Unlike first-order optimization, ZO estimates updates only from final-loss differences caused by random perturbations. Therefore, a layer can contribute more to ZO fine-tuning when its perturbation has a stronger effect on the forward computation. We find that the dominant layer satisfies this condition because it appears early in the model and aligns with the first activation-outlier layer. Perturbations at this layer enter the residual stream and affect the activations of all remaining layers. This propagation allows the perturbation effect to be repeatedly transformed and accumulated before reaching the final loss, leading to larger final-loss changes and a stronger loss-difference signal for ZO updates.

This finding of a dominant layer in ZO fine-tuning has both practical and conceptual implications. Practically, this finding suggests that updating only a single layer can be sufficient to achieve performance comparable to full-model MeZO. More importantly, we hope this finding provides insight for future ZO method design, such as explicitly considering where useful updates arise across layers or making updates to non-dominant layers more effective.

Our contributions can be summarized as follows:

- We discover a dominant-layer phenomenon in ZO fine-tuning: tuning a single layer can recover, and sometimes exceed, the performance of full-model ZO fine-tuning.
- We show that the dominant layer is task-agnostic but model-specific, and can be efficiently identified before training using the first activation-outlier layer.
- We explain why the dominant layer learns well under ZO fine-tuning: residual connection propagation amplifies its perturbation effect, leading to larger final-loss changes and stronger ZO update signals.
- We validate the dominant-layer ZO fine-tuning through extensive experiments on two LLMs, LLaMA2-7B and Qwen3-8B, across nine downstream tasks. Compared to MeZO, dominant-layer ZO fine-tuning improves the average score by 0.86% over full-model and 0.61% over LoRA-based [17] ZO fine-tuning. In addition, it achieves a $1.12\times - 4.52\times$ speedup in ZO fine-tuning relative to full-model MeZO.

2 Related Work

Zeroth-order LLM Fine-tuning. Zeroth-order (ZO) optimization estimates update directions from function values rather than explicit backpropagated gradients, using methods such as SPSA, forming its classical foundation [32, 25]. Recently, MeZO [24] first shows that LLMs can be fine-tuned for downstream tasks with inference-level memory, making ZO a memory-efficient alternative to

backpropagation for large models. In practice, MeZO estimates gradients by applying random perturbations to model parameters and measuring the loss difference between two forward passes, without storing intermediate activations for backward propagation. Beyond memory-efficient fine-tuning, forward-only and ZO optimization have also been explored for mitigating catastrophic forgetting in continual learning [13, 40].

To reduce gradient-estimation variance and accelerate convergence for more accurate and efficient fine-tuning, follow-up work mainly improves ZO fine-tuning along three directions. First, one line of work reduces the trainable or perturbed parameter scope through sparse parameter selection [23], transferable static sparsity [15], or random layer-wise sparse updates [36]. Second, another line of work reduces gradient-estimation variance by designing more informative perturbation directions, including low-rank directions in LOZO [6], random subspaces in SubZero [41], activation-derived directions in AGZO [22], and curvature-aware directions in HiZOO [45]. Third, optimizer-level methods modify the update rule to improve optimization speed and stability, including layer-wise calibration in DiZO [34], clipping and annealing in HELENE [44], faster estimators in FZOO [9], and learned update rules in ZO Fine-tuner [42]. AGZO [22] uses activations to guide perturbation directions, while Guo et al. [15] use first-order gradients and Fisher information to select a sparse parameter subset. Our work studies where effective ZO adaptation occurs across layers and identifies a dominant layer using only inference-time activation statistics.

Selective Layer-wise Fine-tuning. Layer-wise fine-tuning and layer-importance analysis have been widely studied in first-order LLM adaptation. Parameter-efficient tuning methods such as adapters and LoRA [16, 17], together with intrinsic-dimensionality analyses [1], suggest that effective adaptation often lies in a smaller update space than full-model fine-tuning implies. Recent methods further exploit layer-wise importance: LISA selectively freezes middle layers [26], ILA identifies layers critical for alignment [30], and IST/OwLore update selected layers based on layer importance or outlier-weighted sampling [39, 21]. In contrast, to the best of our knowledge, our work is the first to systematically analyze how layer-wise fine-tuning behaves under ZO optimization.

Outlier Activations in LLMs. Outlier activation is a common phenomenon in LLMs, first highlighted by LLM.int8() [11] as a unique challenge for model compression: a small number of activation dimensions exhibit extremely large magnitudes compared to the average of the activation distribution. One important property is that these outliers consistently appear in the same activation dimensions across different inputs, suggesting that they come from the model structure rather than from any specific input sample [33, 2]. Following this observation, a series of works study how to address activation outliers for efficient model compression, especially for quantization and pruning. For example, LLM.int8() isolates outlier features for mixed-precision inference [11], while SmoothQuant migrates activation outlier difficulty into weights to enable accurate low-bit quantization [37].

3 Dominant Layer in ZO Fine-Tuning: Discovery and Identification

3.1 Preliminary: Zeroth-Order Optimization

Following the classical two-point SPSA estimator [32] adopted by MeZO for ZO fine-tuning of LLMs [24], we estimate gradients from two perturbed loss evaluations. At iteration t with parameters θ_t and minibatch $\mathcal{B}_t$, we sample random perturbation vector z_t and compute the ZO gradient estimate as:

$$\widehat{g}_t = \frac{\mathcal{L}(\theta_t + \epsilon z_t; \mathcal{B}_t) - \mathcal{L}(\theta_t - \epsilon z_t; \mathcal{B}_t)}{2\epsilon} z_t, \tag{1}$$

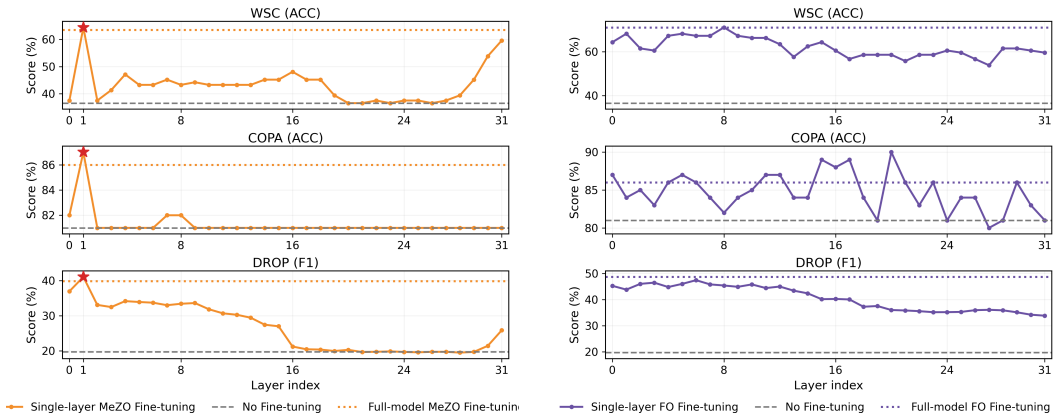
and the parameter update is

$$\theta_{t+1} = \theta_t - \eta_t \widehat{g}_t, \tag{2}$$

where ϵ is the perturbation scale and η_t is the learning rate.

3.2 Empirical Discovery: A *Dominant Layer* Exists in ZO Fine-Tuning

We begin by analyzing how ZO fine-tuning behaves across layers by isolating each layer's contribution. Specifically, based on MeZO, we fine-tune one layer at a time while freezing all other layers, using the same ZO update configuration for every layer. We conduct this study on LLaMA2-7B [35]



(a) MeZO layerwise results. (b) FO layerwise results.

Figure 1: Layer-wise fine-tuning on LLaMA2-7B results across three representative datasets.

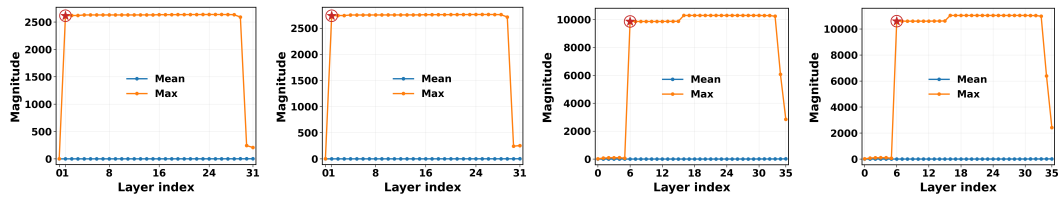
across multiple tasks, including WSC [20], COPA [29], and DROP [12], which cover classification, multiple-choice, and generation settings.

As shown in Figure 1, we make two key observations:

(1) ZO fine-tuning is highly uneven across layers. Performance varies significantly across layers, and most layers provide little or no improvement over the no fine-tuning baseline. For example, on the COPA dataset, only a small subset of layers (4 out of 32 in LLaMA2-7B) improve accuracy after fine-tuning, while the majority remain close to the baseline.

(2) A *dominant layer* clearly emerges in ZO fine-tuning. One specific layer achieves substantially higher performance than other layers and matches, or even exceeds, full-model MeZO fine-tuning. We refer to this layer as the **dominant layer**. Moreover, we find that this layer has two important properties. First, it is **task-agnostic**: for a given LLM model, the same layer consistently dominates across different tasks. For example, in LLaMA2-7B, layer 1 achieves the best performance on all three tasks. Second, it appears to be **model-specific**: different model families may have different dominant-layer indices. For example, the dominant layer is layer 1 in LLaMA2-7B and layer 6 in Qwen3-8B, with the Qwen3-8B layer-wise analysis provided in Appendix B.

To further examine whether this behavior is specific to ZO, we repeat the same layer-wise analysis using first-order fine-tuning. As shown in Figure 1b, first-order fine-tuning exhibits a different pattern: most layers achieve clear improvements over the baseline, and no single layer consistently dominates. This contrast indicates that fine-tuning effectiveness is more concentrated across layers under ZO than under FO in these experiments.



(a) LLaMA2-7B on COPA. (b) LLaMA2-7B on WSC. (c) Qwen3-8B on COPA. (d) Qwen3-8B on WSC.

Figure 2: Mean and Maximum of Layer-wise output activation magnitudes for LLaMA2-7B and Qwen3-8B on COPA and WSC training samples. The highlighted point indicates the first activation-outlier layer, where the maximum output activation magnitude shows a clear jump.

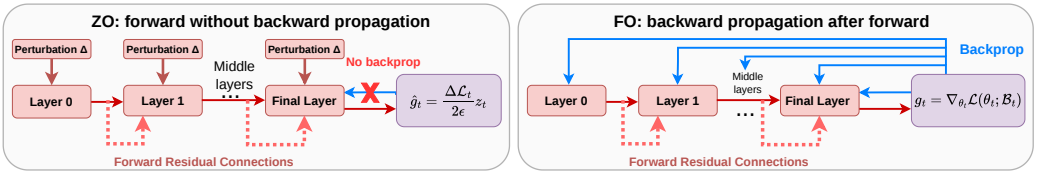
3.3 Identifying the *Dominant Layer* via Activation Outliers

The layer-wise analysis above identifies the dominant layer, but it requires fine-tuning each layer separately, which is computationally expensive. We therefore ask whether the dominant layer can be identified before fine-tuning, using only lightweight signals from the pre-trained model.

Our key observation is that the dominant layer aligns with the first layer where activation outliers emerge. This is motivated by the activation-outlier phenomenon: activation outliers appear consistently at specific activation dimensions and are agnostic to input data, which exhibits a similar property to the dominant layer observed in Section 3.2. As shown in Figure 2, the first activation-outlier layer coincides with the dominant layer identified by layer-wise ZO fine-tuning. For example, the first outlier layer appears at layer 1 in LLaMA2-7B and layer 6 in Qwen3-8B, matching the dominant layers identified by layer-wise analysis.

Based on this observation, we propose a simple inference-only selection method. Given a small calibration set, we run the pre-trained model forward and compute layer-wise activation statistics. Following the activation-outlier criterion of An et al. [2], we select the earliest layer whose maximum-to-mean activation magnitude ratio exceeds 1000. This method identifies the dominant layer before training and avoids expensive layer-wise ZO fine-tuning.

4 Why Does the *Dominant Layer* Emerge in ZO Fine-Tuning?



(a) Zeroth-Order Optimization (ZO) (b) First-Order Optimization (FO)

Figure 3: Comparison of optimization signal flow in ZO and FO. In ZO, gradients are estimated only from forward. In FO, backpropagation provides exact gradients throughout the network.

ZO fine-tuning relies entirely on changes in the final loss under parameter perturbations. As a result, layers differ in effectiveness based on how strongly their perturbations influence the final loss, as illustrated in Figure 3(a). In contrast, FO fine-tuning distributes gradients across layers through backpropagation, preventing such concentration, as shown in Figure 3(b). To evaluate how each layer contributes to the final loss change, we perform a perturbation-only sensitivity analysis before ZO fine-tuning, where we apply random perturbations to one layer at a time and measure the resulting change in final loss. Here, perturbation sensitivity refers to the change in the final loss induced by parameter perturbations of the same scale. As shown in Figure 4, the dominant layer exhibits a strong perturbation-induced loss response, consistent with its effectiveness in layer-wise ZO fine-tuning. In contrast, most other layers induce only minor loss change, which explains their limited contribution to fine-tuning performance.

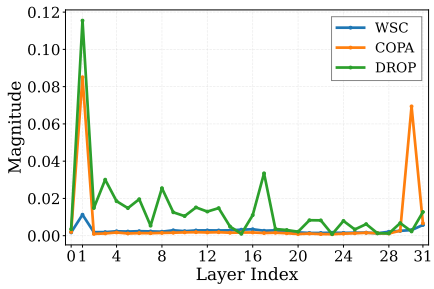


Figure 4: Layerwise Loss Change under 1-step Perturbation on LLaMA2-7B across 3 tasks.

However, the magnitude of the change in loss alone does not fully explain the dominant-layer phenomenon. For example, Figure 4 also shows that some later layers, such as layer 30, can produce noticeable loss changes under perturbation. Nevertheless, these layers do not achieve comparable ZO fine-tuning performance as shown in Figure 1a. This suggests that a large perturbation-induced loss change alone does not ensure effective ZO fine-tuning. This observation motivates considering both perturbation sensitivity and the position at which perturbations enter the residual stream.

In contrast, due to residual connections between layers in LLMs, the output activation of an earlier layer can affect remaining layers and accumulate into the final hidden activation, as illustrated in

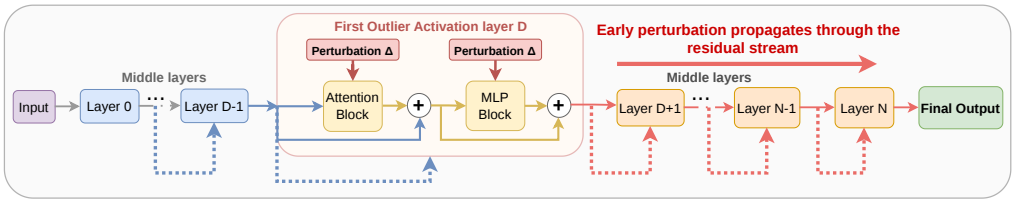


Figure 5: Schematic view of how ZO perturbations are injected at the first outlier-activation layer and propagate through later decoding layers via the residual stream.

Figure 5. This effect is especially important for the dominant layer because it aligns with the first activation-outlier layer, whose output contains extremely large-magnitude activations. Once the perturbation effect from the dominant layer enters the following layers, it can affect their hidden activations and continue to accumulate toward the final activation. Therefore, the resulting loss change is not only large in magnitude, but also comes from a propagated effect across all remaining layers. This propagation helps explain why perturbations at the dominant layer provide an effective signal for ZO fine-tuning. Figure 6 further supports this explanation from the training trajectory. Across both LLaMA2-7B and Qwen3-8B, the dominant layer reduces training loss faster than later outlier layers and follows a trajectory closer to full-model fine-tuning. Although later layers also contain activation outliers, their loss decreases much more slowly, suggesting that activation outliers alone are insufficient; early residual-stream propagation is also important for effective ZO fine-tuning.

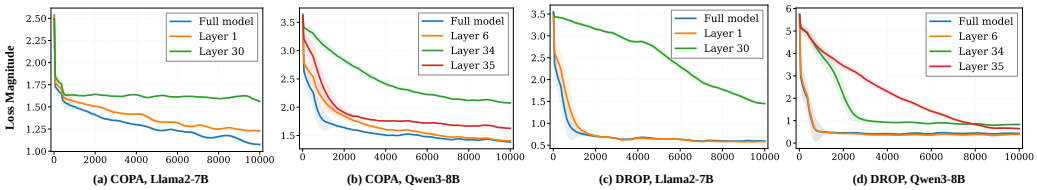


Figure 6: Training loss curve comparison. In both LLaMA2-7B and Qwen3-8B, the dominant layer reduces training loss more effectively than later activation outlier layers.

5 Experimental Validation of Dominant Layer in ZO Fine-tuning

5.1 Experimental Setting

Models and Datasets. To evaluate performance, we conduct experiments on LLaMA2-7B [35] and Qwen3-8B [38] over classification, multiple-choice, and generation tasks used in MeZO [24], including SST-2 [31], RTE [8, 3, 14, 4], CB [10], BoolQ [7], WSC [20], MultiRC [19], COPA [29], SQuAD [28], and DROP [12]. We also evaluate Mistral-7B [18] on SST-2, WSC, COPA, and DROP.

Comparison Setup. To assess whether a single dominant layer can capture the useful adaptation achieved by full-model ZO fine-tuning, we first compare dominant-layer ZO against full-model MeZO [24] and MeZO LoRA, the most common PEFT variant of MeZO. We also include zero-shot inference without fine-tuning and full-model first-order fine-tuning with AdamW as references, allowing us to understand the gap between ZO and FO fine-tuning. In addition, we compare with Sparse-MeZO [23], which reduces the perturbation parameters through sparse masking. We also compare with AGZO [22], which focuses on improving the perturbation direction. Dominant-layer ZO uses the same ZO-SGD update rule as MeZO, but restricts both perturbations and updates to the identified dominant layer. To ensure a controlled comparison, all methods follow the MeZO training and evaluation protocol. We tune the learning rate for all methods and the perturbation scale for ZO methods. Full hyperparameter ranges, training steps and other implementation details are provided in the Appendix. Unless stated, we use random seed 0 for experiments.

Table 1: Performance of fine-tuning LLaMA2-7B (with 1000 examples). FT: full fine-tuning.

Task	SST-2	RTE	CB	BoolQ	WSC	MultiRC	COPA	SQuAD	DROP	AVG.
Task Type	classification						multiple choice	generation		
Zero-shot w/o finetune	58.02	61.73	32.14	66.7	36.54	45.3	81	60.43	19.73	51.29
First Order Adamw FT	95.87	84.84	85.71	86.7	71.15	82.6	86	90.68	48.74	81.37
MeZO FT	92.32	65.34	71.43	76.7	63.46	64.4	86	87.32	39.85	71.87
MeZO LoRA	92.66	65.89	69.64	77.6	63.46	65.9	88	86.26	40.57	72.22
Dominant-layer ZO FT	90.79	67.51	69.64	76.5	64.42	65.86	87	89.2	41.05	72.44
Dominant-layer ZO LoRA	91.02	67.44	67.86	77.8	62.5	66.52	87	88.58	42.10	72.31

5.2 Main Results

Dominant-layer ZO matches or sometimes exceeds full-model ZO performance. Across both model families, restricting ZO updates to a single dominant layer achieves performance comparable to, and sometimes better than, full-model MeZO. On LLaMA2-7B, dominant-layer ZO FT improves over full-model MeZO FT by an average gain of 0.57%, with larger task-level gains on RTE ($+2.17\%$). Dominant-layer ZO LoRA also improves over MeZO LoRA by 0.09% on average, with notable gains on SQuAD ($+2.32\%$). The gains are higher on Qwen3-8B: dominant-layer MeZO improves over full-model MeZO by 1.15% on average, with the largest improvements on CB ($+3.57\%$) and SST-2 ($+2.04\%$). Dominant-layer MeZO LoRA similarly improves over MeZO LoRA by 1.12% on average, with strong gains on CB ($+3.57\%$) and COPA ($+2\%$). Section 5.3 presents additional results across random seeds and on Mistral-7B.

Table 2: Performance of fine-tuning Qwen3-8B (with 1000 examples). FT: full fine-tuning.

Task	SST-2	RTE	CB	BoolQ	WSC	MultiRC	COPA	SQuAD	DROP	AVG
Task type	classification						multiple choice	generation		
Zero-shot w/o finetune	58.03	87	82.14	78.3	70.19	76.4	82	82.97	63.55	75.62
First Order Adamw FT	95.41	92.06	94.64	89.9	78.85	90.1	89	93.65	71.63	88.36
MeZO FT	92.11	90.25	92.86	85.0	70.19	87.2	89	89.98	64.12	84.52
MeZO LoRA	91.74	90.75	91.07	85.5	71.15	86.8	87	90.41	64.59	84.33
Dominant-layer MeZO	94.15	91.34	96.43	84.7	72.12	85.6	90	90.76	65.94	85.67
Dominant-layer MeZO LoRA	92.89	91.53	94.64	86.9	73.08	85.49	89	90.74	64.85	85.46

Comparison with Sparse-MeZO. Sparse-MeZO [23] selects low-magnitude weights and restricts the ZO perturbation to this selected weight subset to make ZO fine-tuning more stable. It constructs a binary mask and replaces the dense MeZO perturbation with a masked perturbation, so only the selected weights are updated. Following the default settings, the smallest magnitude selection threshold is set at 25% before training, while the mask can be regenerated during training by comparing the current parameters with the fixed threshold. Table 3 compares Sparse-MeZO with our dominant-layer ZO on LLaMA2-7B. Sparse-MeZO improves the nine-task average performance by 0.54% over MeZO FT. Dominant-layer MeZO improves 0.57% over MeZO FT and 0.03% over Sparse-MeZO. This indicates that choosing where to apply ZO at the layer level can be as important as choosing parameters within every tensor.

Table 3: Comparison with Sparse-MeZO on LLaMA2-7B. FT: full fine-tuning.

Method	SST-2	RTE	CB	BoolQ	WSC	MultiRC	COPA	SQuAD	DROP	AVG
MeZO FT	92.32	65.34	71.43	76.70	63.46	64.40	86.00	87.32	39.85	71.87
Sparse-MeZO	91.63	67.90	71.43	77.50	63.46	63.20	86.00	88.83	41.72	72.41
Dominant-layer MeZO	90.79	67.51	69.64	76.50	64.42	65.86	87.00	89.20	41.05	72.44

Comparison with AGZO. AGZO [22] uses activation-guided low-rank perturbations within linear modules to improve the perturbation direction. Table 4 compares AGZO with full-model and dominant-layer MeZO on SST-2, WSC, and COPA for LLaMA2-7B and Qwen3-8B. On LLaMA2-

7B, dominant-layer MeZO improves the three-task average performance by 0.15% over MeZO FT and 0.96% over AGZO. On Qwen3-8B, the corresponding gains are 1.65% and 1.81%. This indicates that choosing where to apply ZO needs to be considered while constraining the perturbation direction.

Table 4: AGZO comparison on SST-2, WSC, and COPA.

Model	Method	SST-2	WSC	COPA	AVG
LLaMA2-7B	MeZO FT	92.32	63.46	86.00	80.59
	AGZO	91.97	65.38	82.00	79.78
	Layer-1 MeZO	90.79	64.42	87.00	80.74
Qwen3-8B	MeZO FT	92.11	70.19	89.00	83.77
	AGZO	94.72	72.12	84.00	83.61
	Layer-6 MeZO	94.15	72.12	90.00	85.42

5.3 Consistency Across Random Seeds and Models

Results across random seeds. To assess whether the comparable performance of dominant-layer MeZO persists across random seeds, we repeat full-model and dominant-layer MeZO with three perturbation seeds (0, 42, and 2026) on SST-2, WSC, COPA, and DROP for LLaMA2-7B and Qwen3-8B, keeping other settings fixed. Table 5 reports mean $\pm$ standard deviation. Dominant-layer MeZO maintains comparable mean performance on this four-task subset, supporting the observation beyond a single run.

Table 5: Results across three random seeds on four tasks. Task scores are mean $\pm$ standard deviation. DROP reports F1; the other tasks report accuracy.

Model	Method	SST-2	WSC	COPA	DROP	AVG
LLaMA2-7B	MeZO FT	92.43 ± 0.20	62.50 ± 1.66	85.33 ± 0.58	39.04 ± 0.95	69.83
	Layer-1 MeZO	90.33 ± 0.52	63.14 ± 1.47	86.33 ± 0.58	40.50 ± 1.19	70.08
Qwen3-8B	MeZO FT	91.25 ± 1.66	70.83 ± 0.55	88.00 ± 1.00	64.37 ± 0.88	78.61
	Layer-6 MeZO	92.70 ± 2.42	71.47 ± 0.56	88.67 ± 1.15	66.11 ± 0.42	79.74

Results on Mistral-7B. To examine whether the dominant-layer observation extends to another model family, we evaluate Mistral-7B on the same four tasks. The same inference-only activation analysis identifies layer 1, which we use across all four tasks. Table 6 shows that layer-1 MeZO outperforms full-model MeZO by 0.76 percentage points on average across the four tasks.

Table 6: Mistral-7B results on four tasks. DROP reports F1; the other tasks report accuracy.

Method	SST-2	WSC	COPA	DROP	AVG
Zero-shot	55.62	37.50	85.00	28.70	51.71
MeZO FT	92.66	55.77	85.00	43.73	69.29
Layer-1 MeZO	92.78	54.81	86.00	46.60	70.05

5.4 Training Efficiency

Dominant-layer ZO fine-tuning also improves training efficiency. Each MeZO step consists of full-model forward evaluations, parameter perturbation, and parameter update. Since the perturbed losses still require full-model forward passes, restricting the optimized parameters does not substantially reduce the forward cost. Therefore, the expected efficiency gain comes mainly from the parameter-side operations when using Dominant-layer ZO. For LLaMA2-7B, which has 32 decoding layers, updating only one dominant layer gives an ideal parameter-side reduction of about $32\times$ compared with full-model MeZO. The measured results in Table 7 closely match this expectation: single-layer ZO reduces perturbation time by 27.34–$31.28\times$ and update time by 31.56–$32.71\times$ across tasks. Thus, dominant-layer ZO realizes nearly the full theoretical saving for parameter perturbation and update. The end-to-end speedup depends on the fraction of the forward pass. On short-input tasks, where

perturbation and update dominate a larger portion of each step, dominant-layer ZO achieves larger improvements, such as $4.52\times$ on COPA and $2.45\times$ on SST-2. On long-input tasks, the full-model forward pass dominates the runtime, so the total speedup is smaller, such as $1.24\times$ on CB and $1.12\times$ on DROP. This confirms that dominant-layer ZO is most beneficial when parameter-side operations are a significant bottleneck.

We also compare with Sparse-MeZO as an elementwise sparse perturbation baseline. Although Sparse-MeZO keeps fewer than 25% of parameters active in our setting, its perturbation and update speedups are much smaller. This is expected because Sparse-MeZO applies scattered masks within tensors, while dominant-layer ZO preserves dense contiguous tensor operations on a single decoding layer. Overall, dominant-layer ZO provides stronger practical speedups by combining a layer-level reduction in updated parameters with efficient dense computation.

Table 7: Per-step runtime breakdown for SST2, CB, WSC, COPA, and DROP under different ZO settings on LLaMA2-7B. (Speedups) are relative to full-model MeZO for the same task and runtime component. For Sparse-MeZO, we exclude dynamic mask construction time to isolate the cost of the core ZO perturbation and update operations.

Task	Param Range	Forward/step	Perturb/step	Update/step	Total/step
SST2	Full model	0.4431s (1.00x)	0.4849s (1.00x)	0.2061s (1.00x)	1.1341s (1.00x)
SST2	Sparse-MeZO	0.4454s (0.99x)	0.3714s (1.31x)	0.1317s (1.57x)	0.9485s (1.20x)
SST2	Dominant layer	0.4411s (1.00x)	0.0155s (31.28x)	0.0063s (32.71x)	0.4629s (2.45x)
CB	Full model	2.9190s (1.00x)	0.4867s (1.00x)	0.2061s (1.00x)	3.6118s (1.00x)
CB	Sparse-MeZO	2.8469s (1.03x)	0.3842s (1.27x)	0.1365s (1.51x)	3.3676s (1.07x)
CB	Dominant layer	2.8815s (1.01x)	0.0178s (27.34x)	0.0065s (31.71x)	2.9057s (1.24x)
WSC	Full model	1.0724s (1.00x)	0.4858s (1.00x)	0.2083s (1.00x)	1.7665s (1.00x)
WSC	Sparse-MeZO	1.0623s (1.01x)	0.3775s (1.29x)	0.1301s (1.60x)	1.5699s (1.13x)
WSC	Dominant layer	1.0766s (1.00x)	0.0161s (30.17x)	0.0066s (31.56x)	1.0992s (1.61x)
COPA	Full model	0.1764s (1.00x)	0.4856s (1.00x)	0.2081s (1.00x)	0.8701s (1.00x)
COPA	Sparse-MeZO	0.1655s (1.07x)	0.3490s (1.39x)	0.1242s (1.68x)	0.6387s (1.36x)
COPA	Dominant layer	0.1696s (1.04x)	0.0163s (29.79x)	0.0064s (32.52x)	0.1923s (4.52x)
DROP	Full model	5.8392s (1.00x)	0.4852s (1.00x)	0.2084s (1.00x)	6.5328s (1.00x)
DROP	Sparse-MeZO	5.9472s (0.98x)	0.3707s (1.31x)	0.1334s (1.56x)	6.4512s (1.01x)
DROP	Dominant layer	5.8273s (1.00x)	0.0158s (30.71x)	0.0066s (31.58x)	5.8496s (1.12x)

Additional measurements of peak GPU memory, optimizer-state storage, and trainable-parameter storage are reported in Appendix D.2.

5.5 Ablation Study and Analysis

Does combining layers improve over the dominant layer? We next study whether the dominant layer can be further improved by adding another layer for tuning. As an example, we select layer 30, which also shows relatively large loss changes under perturbation. Table 8 shows that simply adding another layer does not reliably improve performance. On SST-2, tuning layer 1 together with layer 30 slightly improves over tuning the dominant layer alone, but the gain remains close to full-model MeZO. On COPA, however, the two-layer setting does not improve over the dominant layer. These task-dependent changes suggest that the benefit of adding a layer depends on how its updates interact with those of the dominant layer.

Table 8: Layer-combination results on LLaMA2-7B. FT: full fine-tuning.

Setting	SST-2	COPA
w/o Finetune	58.02	81
MeZO FT	92.32	86
Dominant-layer MeZO	90.79	87
Dominant Layer + Layer 30	91.52	86

Impact of components within the dominant layer. To examine the contributions of different components, we first compare attention-only and MLP-only tuning within the dominant layer (Table 9). MLP-only tuning performs better than attention-only tuning on both tasks.

Table 9: MeZO component ablations on LLaMA2-7B. All component and channel restrictions refer to the dominant layer (layer 1).

Setting	WSC	COPA
Zero-shot	36.54	81
MeZO FT	63.46	86
Layer-0 MeZO	37.5	82
Full dominant-layer MeZO	64.42	87
Attention-only	37.5	85
MLP-only	62.5	86
MLP top 1% outlier channels	62.5	83
Dominant layer excluding top 1% MLP outlier channels	56.73	81
MLP excluding top 1% MLP outlier channels	37.5	81

Motivated by prior studies showing that activation outliers occur in a small number of fixed feature dimensions and are closely related to the MLP down-projection layer [33, 2], we further examine the corresponding channels inside the dominant layer. Specifically, since these outlier dimensions mainly connect to the MLP module within a decoder layer, we study whether the dominant-layer advantage is driven by the associated MLP channels. Table 9 shows that these activation-outlier MLP channels play a critical role. Fine-tuning only the dominant-layer MLP recovers most of the full dominant-layer performance, and tuning only the top 1% activation-outlier MLP channels still preserves much of the gain on WSC. In contrast, removing these channels from MLP tuning causes performance to collapse close to the no-fine-tuning baseline, indicating that they provide a major part of the ZO update signal. At the same time, the dominant-layer effect cannot be fully reduced to this small channel subset. Removing the top 1% outlier channels from full dominant-layer tuning significantly reduces performance, especially on COPA, but does not completely remove the gain on WSC. These results suggest that activation-outlier MLP channels serve as high-leverage components for ZO adaptation, while the rest of the dominant layer still provides additional adaptation capacity.

Perturbation scale and retained updates. To assess whether non-dominant layers benefit from stronger perturbations, we increase the perturbation scale for representative early, middle, and late layers on LLaMA2-7B WSC. Larger scales do not close their performance gap to the dominant layer (Appendix C.2). We also examine updates learned during full-model MeZO fine-tuning by retaining one layer's updates and restoring all other layers to their pretrained weights. The best layer under this keep-only evaluation varies by task and can differ from the independently trained dominant layer (Appendix C.3).

6 Conclusion

We study how the effectiveness of zeroth-order fine-tuning varies across LLM layers and identify a single dominant layer. Across two model families and nine downstream tasks, tuning this layer often matches or exceeds full-model MeZO, while matched first-order fine-tuning shows much weaker layer concentration. We further show that this layer aligns with the first activation-outlier layer, enabling inference-only identification before training. Our analysis suggests that the dominant layer combines high perturbation sensitivity with an early position in the residual stream, allowing its perturbation to affect many subsequent blocks. This produces a stronger forward-loss signal under ZO, which relies on loss differences rather than backpropagated gradients. Overall, our results show that the effectiveness of ZO fine-tuning varies substantially across layers, motivating methods that account for layer identity and within-layer importance.

Limitation. There is still a performance gap between Dominant-layer ZO and first-order fine-tuning methods. Dominant-layer ZO still takes many steps to achieve good performance, which remains a problem for applications. Extending the study to a broader range of model architectures and scales, and combining dominant-layer tuning with other ZO optimizers such as ZO-AdaMM [5] and FZOO [9], remain directions for future work.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Science Foundation under Grant Nos. 2505326, 2528767, and 2348376, and by a seed grant from the School of Data Science at the University of North Carolina at Charlotte. The authors declare no competing financial interests. We thank the Department of Computer Science at Texas Christian University for its support.

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328, 2021.

[2] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic outliers in large language models. In *International Conference on Learning Representations*, pages 31483–31504, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4e23bee2df5fc72018d9d74d875d7cf3-Paper-Conference.pdf.

[3] R Bar-Haim, I Dagan, B Dolan, L Ferro, D Giampiccolo, B Magnini, and I Szpektor. The second pascal recognising textual entailment challenge. In *The Second PASCAL Recognising Textual Entailment Challenge*, 2006.

[4] Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST, 2009. URL https://tac.nist.gov/publications/2009/additional.papers/RTE5_overview.proceedings.pdf.

[5] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David D. Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. In *NeurIPS*, pages 7202–7213, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/576d026223582a390cd323bef4bad026-Abstract.html.

[6] Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. In *International Conference on Learning Representations*, pages 62581–62607, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/9ccc9d814d3dee4750debaf23061e733-Paper-Conference.pdf.

[7] 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.

[8] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.

[9] Sizhe Dang, Yangyang Guo, Yanjun Zhao, Xiaodong Zheng, Guang Dai, Ivor Tsang, and Haishan Ye. Fzoo: Fast zeroth-order optimizer for fine-tuning large language models towards adam-scale speed. In *International Conference on Learning Representations*, pages 113760–113792, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b8d4c82b365d9361863000f8d1e86f1b-Paper-Conference.pdf.

[10] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124, 2019.

[11] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc., 2022.

[12] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, 2019.

[13] Tao Feng, Wei Li, Didi Zhu, Hangjie Yuan, Wendi Zheng, Dan Zhang, and Jie Tang. ZeroFlow: Overcoming catastrophic forgetting is easier than you think. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 16808–16824. PMLR, 13–19 Jul 2025. URL `https://proceedings.mlr.press/v267/feng25j.html`.

[14] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

[15] Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, et al. Zeroth-order fine-tuning of llms with transferable static sparsity. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[18] 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`.

[19] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, 2018.

[20] Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. *KR*, 2012(13th):3, 2012.

[21] Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. Outlier-weighed layerwise sampling for LLM fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19460–19473. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.996.

[22] Wei Lin, Yining Jiang, Qingyu Song, Qiao Xiang, and Hong Xu. AGZO: Activation-guided zeroth-order optimization for LLM fine-tuning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, pages 74416–74438. PMLR, 2026. URL `https://proceedings.mlr.press/v306/lin26n.html`.

[23] Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse mezo: Less parameters for better performance in zeroth-order llm fine-tuning. In *Advances in Neural Information Processing Systems*, pages 21139–21169. Curran Associates, Inc., 2025. doi: 10.52202/085713-0712.

[24] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[25] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[26] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37:57018–57049, 2024.

[27] Xin Qiu, Yulu Gan, Conor F. Hayes, Qiyao Liang, Yinggan Xu, Roberto Dailey, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: LLM fine-tuning beyond reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 101987–102015. PMLR, 2026. URL `https://proceedings.mlr.press/v306/qiu26f.html`.

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392, 2016.

[29] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95, 2011.

[30] Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, and Xiao-Ming Wu. Understanding layer significance in llm alignment, 2025.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[32] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[33] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

[34] Qitao Tan, Jun Liu, Zheng Zhan, Caiwen Ding, Yanzhi Wang, Xiaolong Ma, Jaewoo Lee, Jin Lu, and Geng Yuan. Harmony in divergence: Towards fast, accurate, and memory-efficient zeroth-order llm fine-tuning. *Advances in Neural Information Processing Systems*, 38:175003–175037, 2025.

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

[36] Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. Simultaneous computation and memory efficient zeroth-order optimizer for fine-tuning large language models, 2024.

[37] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

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

[39] Kai Yao, Penglei Gao, Lichun Li, Yuan Zhao, Xiaofeng Wang, Wei Wang, and Jianke Zhu. Layer-wise importance matters: Less memory for better performance in parameter-efficient fine-tuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1977–1992, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[40] Wanhao Yu, Zheng Wang, Shuteng Niu, Sen Lin, and Li Yang. More than memory savings: zeroth-order optimization mitigates forgetting in continual learning. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4766–4776. IEEE, 2026.

[41] Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of llms in random subspaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4475–4485, 2025.

[42] Kairun Zhang, Haoyu Li, Yanjun Zhao, Yifan Sun, and Huan Zhang. Learning a zeroth-order optimizer for fine-tuning LLMs. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 156874–156894. PMLR, 2026. URL https://proceedings.mlr.press/v306/zhang26da.html.

[43] Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D. Lee, Wotao Yin, Mingyi Hong, Zhangyang Wang, Sijia Liu, and Tianlong Chen. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 59173–59190. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24ad.html.

[44] Huaqin Zhao, Jiaxi Li, Yi Pan, Shizhe Liang, Xiaofeng Yang, Fei Dou, Tianming Liu, and Jin Lu. HELENE: Hessian layer-wise clipping and gradient annealing for accelerating fine-tuning LLM with zeroth-order optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26044–26067. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1323.

[45] Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*, pages 43496–43520, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/6bf82cc56a5fa0287c438baa8be65a70-Paper-Conference.pdf.

A Experimental Details

A.1 Tasks, Models, and Metrics

Following MeZO [24], we construct each task split by sampling up to 1000 training examples, 500 validation examples, and 1000 evaluation examples when sufficient data are available. For smaller datasets, we follow the same protocol but reduce the validation split accordingly; in particular, for WSC, CB, and COPA we use a validation set of 100 examples. We evaluate two model families, LLaMA2-7B and Qwen3-8B, over classification, multiple-choice, and generation tasks. The additional Mistral-7B evaluation covers SST-2, WSC, COPA, and DROP under the same MeZO protocol. The Countdown ES extension uses a separate setup, specified in Appendix B.3. ZO fine-tuning uses float16 for all three models, while FO fine-tuning on LLaMA2-7B and Qwen3-8B uses bfloat16. FO experiments are conducted on NVIDIA H200 GPUs. ZO experiments use A6000-48GB GPUs for LLaMA2-7B and NVIDIA H200 GPUs for Qwen3-8B and Mistral-7B.

Unless otherwise stated, we keep the task formatting, data budget, and evaluation protocol fixed across methods so that the comparisons isolate the effect of the optimization method or trainable scope. In particular, the comparisons among full-model ZO, dominant-layer ZO, Sparse-MeZO, FO fine-tuning, and MeZO-LoRA use the same prompt family and validation-based model selection procedure. This shared setup is important for interpreting the layerwise results: differences in performance should be attributed to optimization behavior rather than to prompt or data changes.

Activation-based layer identification. We collect layer-output activation magnitudes during inference with the pretrained model and use the maximum-to-mean magnitude ratio described in Section 3.3. The earliest layer with a ratio above 1000 is selected before fine-tuning, following the empirical outlier convention of An et al. [2]. Figure 2 shows the output activation statistics on COPA and WSC training samples. The selected indices are layer 1 for LLaMA2-7B, layer 6 for Qwen3-8B, and layer 1 for Mistral-7B; the selected layer is then held fixed across the evaluated tasks for each model. In our experiments, using 1 calibration example, 10 examples, or the entire training set yields the same selected layer for each model. For Mistral-7B, we validate this activation-based choice by comparing layer-1 MeZO with full-model MeZO on four tasks.

A.2 Hyperparameters

Tables 10 and 11 summarize the hyperparameter grids used in our LLaMA2-7B and Qwen3-8B experiments. For MeZO-style methods, we use constant learning rates, a fixed perturbation scale and 10k steps, while FO fine-tuning with AdamW follows a separate learning-rate grid over 5 epochs. For all methods, we select the final checkpoint based on the lowest validation loss among checkpoints saved every 2k training steps. Following the setting from Subzero [41], we use default sparse rate 0.75 for Sparse-MeZO across all datasets. For AGZO on LLaMA2-7B and Qwen3-8B, we use the corresponding model's MeZO hyperparameter settings in Tables 10 and 11.

For both full-model and dominant-layer MeZO on Mistral-7B, we search learning rates over $\{5 \times 10^{-8}, 1 \times 10^{-7}, 5 \times 10^{-7}, 1 \times 10^{-6}\}$ and fix the perturbation scale at $\epsilon = 0.001$.

The hyperparameter tables also clarify an important aspect of our comparisons: the dominant-layer advantage is not due to using a more favorable optimization budget for the restricted scope. Instead, dominant-layer ZO inherits essentially the same MeZO optimization protocol as the full-model baseline, differing only in which parameters are perturbed and updated. This makes the dominant-layer results in the main paper a structural finding rather than a consequence of hyperparameter tuning.

A.3 Prompts

Table 12 lists the prompt templates used in our experiments. We follow MeZO [24] for templates and keep the prompt template fixed across zero-shot evaluation, FO fine-tuning, and ZO fine-tuning for each task. This consistency is especially important for the layerwise analyses, since it ensures that the observed differences across layers and optimization methods are not confounded by changes in verbalization or answer formatting.

Table 10: The hyperparameter grids used for LLaMA2-7B experiments. All weight decay is set to 0. FO FT uses 5 epochs and MeZO uses 10K steps and constant learning rates. We check validation performance and save the best checkpoint every 2k total training steps.

Experiment	**Hyperparameters**	**Values**
MeZO FT	Batch size	16
	Learning rate	$\{1e{-}7, 5e{-}7, 1e{-}6, 5e{-}6\}$
	ϵ	$1e{-}3$
MeZO Single Layer	Batch size	16
	Learning rate	$\{1e{-}7, 5e{-}7, 1e{-}6, 5e{-}6\}$
	ϵ	$1e{-}3$
MeZO (LoRA)	Batch size	16
	Learning rate	$\{5e{-}6, 1e{-}5, 2e{-}5, 5e{-}5\}$
	ϵ	$1e{-}3$
	(r, α)	$(8, 16)$
Sparse-MeZO	Batch size	16
	Learning rate	$\{5e{-}7, 1e{-}6, 2e{-}6, 5e{-}6\}$
	ϵ	$1e{-}3$
	sparse rate	0.75
FO FT with Adamw	Batch size	8
	Learning rates	$\{1e{-}5, 5e{-}5, 1e{-}4\}$

Table 11: The hyperparameter grids used for Qwen3-8B experiments. All weight decay is set to 0. FO FT uses 5 epochs and MeZO uses 10K steps and constant learning rates. We check validation performance and save the best checkpoint every 2k total training steps.

Experiment	**Hyperparameters**	**Values**
MeZO FT	Batch size	16
	Learning rate	$\{1e{-}7, 5e{-}7, 1e{-}6, 5e{-}6\}$
	ϵ	$1e{-}3$
MeZO Single Layer	Batch size	16
	Learning rate	$\{1e{-}7, 5e{-}7, 1e{-}6, 5e{-}6\}$
	ϵ	$1e{-}3$
MeZO (LoRA)	Batch size	16
	Learning rate	$\{5e{-}6, 1e{-}5, 2e{-}5, 5e{-}5\}$
	ϵ	$1e{-}3$
	(r, α)	$(8, 16)$
FO FT with Adamw	Batch size	8
	Learning rates	$\{1e{-}5, 5e{-}5, 1e{-}4\}$

Table 12: Prompt templates used in our experiments. Task types are classification (cls.), multiple-choice (mch.), and question answering (QA). Prompts are same as MeZO [24].

Dataset	Type	Prompt
SST-2	cls.	`<text>` It was terrible/great
RTE	cls.	`<premise>` Does this mean that "`<hypothesis>`" is true? Yes or No? Yes/No
CB	cls.	Suppose `<premise>` Can we infer that "`<hypothesis>`"? Yes, No, or Maybe? Yes/No/Maybe
BoolQ	cls.	`<passage><question>`? Yes/No
WSC	cls.	`<text>` In the previous sentence, does the pronoun "`<span2>`" refer to `<span1>`? Yes or No? Yes/No
MultiRC	cls.	`<paragraph>` Question: `<question>` I found this answer "`<answer>`". Is that correct? Yes or No? Yes/No
COPA	mch.	`<premise>` so/because `<candidate>`
ReCoRD	mch.	`<passage>` `<query>.replace("@placeholder", <candidate>)`
SQuAD	QA	Title: `<title>` Context: `<context>` Question: `<question>` Answer:
DROP	QA	Passage: `<context>` Question: `<question>` Answer:

B Full Layer Sweeps and Optimizer Extension

This section reports additional numerical results that complement the figures and summaries in the main paper. The main paper emphasizes representative plots and high-level comparisons, whereas the appendix provides exact per-task and per-layer values to make the dominant-layer phenomenon fully transparent. Together, these tables show that the concentration of useful ZO adaptation is both numerically sharp and substantially stronger than the corresponding layerwise variation under FO fine-tuning.

B.1 Layer-wise Fine-tuning on Qwen3-8B

Complete 36-layer sweep. We independently fine-tune each of Qwen3-8B's 36 decoder layers under the same MeZO configuration on WSC, COPA, and DROP, covering the three task formats in the main benchmark. All other layers remain frozen in each run. Table 13 reports the complete sweep. The activation-selected layer 6 achieves the highest COPA score (90) and DROP F1 (65.94), and ties for the highest WSC score (72.12) with layers 10, 11, 21, 26, 28, and 29. Thus, the activation-based choice is supported on all three tasks, although WSC alone does not identify a unique best layer. This validation extends the complete layer-wise analysis beyond LLaMA2-7B.

B.2 Detailed Results of FO, ZO Layerwise Fine-tuning on LLaMA2-7B

We next provide the full layerwise FO results for LLaMA2-7B on three representative tasks. These exact values complement the main-text FO figure and make the contrast with ZO more explicit.

Table 14 confirms that FO fine-tuning is heterogeneous across layers, but the pattern is relatively distributed. Many early and middle layers provide substantial gains over the zero-shot baseline, and no single layer consistently accounts for the full-model FO performance across WSC, COPA, and DROP. Although some layers stand out on individual tasks, the overall FO picture is broad rather than sharply concentrated. This numerical pattern supports the main-text claim that the dominant-layer bottleneck is not a generic property of all fine-tuning, but is much sharper under forward-only ZO.

Table 15 reports the corresponding single-layer MeZO ablations on LLaMA2-7B. These exact values provide the numerical counterpart to the main-text layerwise MeZO figure. The contrast with FO is obvious. In Table 15, layer 1 clearly emerges as the dominant layer across WSC, COPA, and

Table 13: Complete 36-layer MeZO sweep Results on Qwen3-8B. WSC and COPA report accuracy; DROP reports F1.

Layer	WSC	COPA	DROP	Layer	WSC	COPA	DROP
0	69.23	82	60.49	18	71.15	83	62.16
1	70.19	82	63.10	19	70.19	82	63.54
2	68.27	82	63.85	20	69.23	83	63.11
3	69.23	84	60.42	21	72.12	83	64.33
4	71.15	83	59.73	22	70.19	83	63.91
5	69.23	83	61.82	23	71.15	82	62.56
6	72.12	90	65.94	24	69.23	82	62.21
7	71.15	85	60.51	25	71.15	82	62.28
8	71.15	82	60.10	26	72.12	82	62.26
9	71.15	86	60.43	27	71.15	82	63.39
10	72.12	85	61.02	28	72.12	82	61.80
11	72.12	83	61.02	29	72.12	82	62.40
12	69.23	84	59.59	30	71.15	82	62.93
13	71.15	83	60.95	31	71.15	82	62.21
14	69.23	83	60.86	32	71.15	82	61.99
15	70.19	84	59.14	33	71.15	82	61.98
16	70.19	86	61.99	34	71.15	83	63.40
17	71.15	87	63.56	35	71.15	83	59.44

Table 14: LLaMA2-7B single-layer FO results across all 32 transformer layers on WSC, COPA, and DROP. WSC and COPA report accuracy (%), while DROP reports F1 (%).

Layers 0–15				Layers 16–31			
Layer	WSC	COPA	DROP F1	Layer	WSC	COPA	DROP F1
0	64.42	87	45.31	16	60.58	88	40.29
1	68.27	84	43.87	17	56.73	89	40.10
2	61.54	85	46.07	18	58.65	84	37.31
3	60.58	83	46.53	19	58.65	81	37.58
4	67.31	86	44.86	20	58.65	90	36.05
5	68.27	87	46.06	21	55.77	86	35.88
6	67.31	86	47.48	22	58.65	83	35.61
7	67.31	84	45.87	23	58.65	86	35.22
8	71.15	82	45.44	24	60.58	81	35.23
9	67.31	84	44.94	25	59.62	84	35.33
10	66.35	85	45.86	26	56.73	84	35.99
11	66.35	87	44.46	27	53.85	80	36.13
12	63.46	87	45.04	28	61.54	81	35.93
13	57.69	84	43.46	29	61.54	86	35.18
14	62.50	84	42.44	30	60.58	83	34.26
15	64.42	89	40.20	31	59.62	81	33.88

Full model w/o finetune: WSC = 36.54, COPA = 81, DROP F1 = 19.73.
FO full model finetune: WSC = 71.15, COPA = 86, DROP F1 = 48.74.

Table 15: LLaMA2-7B single-layer MeZO results across all 32 transformer layers on WSC, COPA, and DROP. WSC and COPA report accuracy (%), while DROP reports F1 (%).

Layers 0–15				Layers 16–31			
Layer	WSC	COPA	DROP F1	Layer	WSC	COPA	DROP F1
0	37.5	82	36.95	16	48.08	81	21.24
1	64.42	87	41.05	17	45.19	81	20.46
2	37.5	81	33.08	18	45.19	81	20.38
3	41.35	81	32.47	19	39.42	81	19.97
4	47.12	81	34.16	20	36.54	81	20.30
5	43.27	81	33.91	21	36.54	81	19.67
6	43.27	81	33.71	22	37.5	81	19.75
7	45.19	82	33.00	23	36.54	81	19.86
8	43.27	82	33.41	24	37.5	81	19.70
9	44.23	81	33.61	25	37.5	81	19.60
10	43.27	81	31.85	26	36.54	81	19.73
11	43.27	81	30.69	27	37.5	81	19.74
12	43.27	81	30.28	28	39.42	81	19.51
13	43.27	81	29.44	29	45.19	81	19.71
14	45.19	81	27.45	30	53.85	81	21.45
15	45.19	81	27.00	31	59.62	81	25.88

Full model w/o finetune: WSC = 36.54, COPA = 81, DROP F1 = 19.73.
MeZO full model finetune: WSC = 63.5, COPA = 86, DROP F1 = 39.85.

DROP, recovering most of the gain of full-model MeZO and even slightly exceeding it on DROP. By comparison, most other layers remain far closer to the zero-shot baseline, especially on COPA and in the later layers on DROP. Although a weaker late-layer rise appears on some tasks, layer 1 remains the strongest independently optimized layer across these three tasks. This comparison concerns independently trained layers and does not locate all useful parameter changes within a full-model MeZO checkpoint.

B.3 Extension to Evolution Strategies

Countdown evaluation. To examine whether the layer-restriction observation extends beyond MeZO's SPSA-based fine-tuning, we evaluate Evolution Strategies (ES) [27] on Countdown with Qwen3-8B. We compare full-model ES against three single-layer settings: layer 6, the first activation-outlier layer; layer 34, a late activation-outlier layer; and layer 0, the first decoder layer. All four settings use a population size of 16, 16 training prompts per population member, 200 optimization iterations, perturbation scale $\sigma = 0.001$, and learning rate $\alpha = 0.0005$. We use greedy decoding and evaluate accuracy on the full 2,000-example test set.

Table 16 shows that layer-6 ES reaches 52.35% accuracy, compared with 58.70% for full-model ES and 29.15% without updating the model. It recovers most of the improvement over the initial model and outperforms the other tested single-layer settings: 35.25% for layer 0 and 32.00% for layer 34. This provides evidence for the usefulness of the first outlier layer in an additional forward-only optimization setting. Layer-6 ES still falls short of full-model ES, and this experiment covers one model and one task rather than the full MeZO benchmark.

Table 16: Evolution Strategies on Countdown with Qwen3-8B. Accuracy is evaluated on the full 2,000-example test set.

Trainable scope	Accuracy (%)
No update	29.15
Full model	58.70
Layer 6 only	52.35
Layer 0 only	35.25
Layer 34 only	32.00

C Controlled Ablations

C.1 Layer Combinations

We compare tuning layer 1 alone with jointly tuning layers 1 and 30 on LLaMA2-7B. Layer 30 is included because it also produces a noticeable loss change under perturbation. Table 8 shows a mixed result: the combination improves SST-2 from 90.79 to 91.52, while COPA decreases from 87 to 86. Adding a layer with high perturbation sensitivity therefore does not consistently improve on the dominant layer. The perturbation sensitivity and early residual-stream placement discussed in Section 4 characterize an individual perturbation location. Layer 1 already combines these factors; adding layer 30 does not change where its perturbation enters the network or extend its downstream propagation path.

C.2 Perturbation-Scale Control

We test whether larger parameter perturbations can improve representative non-dominant layers on LLaMA2-7B WSC. The selected layers are 0, 15, and 30, spanning early, middle, and late positions. Table 17 includes the original $\epsilon = 0.001$ and three larger values, 0.005, 0.01, and 0.05. The best score for layer 0 rises to 39.50 at 0.005, while the original scale gives the best scores for layers 15 and 30 (45.19 and 53.85). None of the tested scales closes the gap to layer 1 at 64.42.

Changing ϵ increases the magnitude of the applied perturbation, but does not change the layer's location or its downstream computation. The observed degradation at larger scales is also consistent with a less local finite-difference approximation. These observations motivate distinguishing perturbation magnitude from layer-wise sensitivity; they do not show that every non-dominant layer or model has been exhaustively tuned.

Table 17: Perturbation-scale control on LLaMA2-7B WSC. The grid contains the original $\epsilon = 0.001$ and three larger scales; scores report accuracy.

Layer	$\epsilon = 0.001$	$\epsilon = 0.005$	$\epsilon = 0.01$	$\epsilon = 0.05$
0	37.5	39.5	37.5	35.58
15	45.19	44.23	44.23	37.5
30	53.85	48.08	47.12	43.27

C.3 Post-hoc Retained Updates

Starting from each full-model MeZO checkpoint, we retain the learned update for one decoder layer and restore all other layers to their pretrained weights. We repeat this operation for every decoder layer. Table 18 summarizes the zero-shot and full-model references, the score obtained by retaining the independently identified dominant layer's update, and the best score across the keep-only sweep with its layer index.

Retaining the dominant layer is best on LLaMA2-7B SST-2, COPA, and DROP, and Qwen3-8B SST-2 and COPA. It is not always best: LLaMA2-7B WSC favors layer 7 (39.42 versus 38.46 for layer 1), Qwen3-8B WSC favors layer 3 (72.12 versus 70.19 for layer 6), and Qwen3-8B DROP favors layer 14 (64.11 versus 63.74 for layer 6). Retaining a single update also need not recover the performance of the full checkpoint, as illustrated by LLaMA2-7B WSC and DROP.

This experiment measures a different property from independent layer-restricted training. In the latter, a layer is optimized while the rest of the model stays fixed throughout training. During full-model MeZO, all layers are instead perturbed and updated jointly using the same scalar loss difference. Their learned changes can depend on changes elsewhere in the model. Post-hoc restoration creates a hybrid of one fine-tuned layer and pretrained layers that was not optimized as a configuration during training. Removing the other updates may break complementary or compensatory changes across layers. This is a possible explanation for the difference between the two experiments, rather than a direct measurement of cross-layer coordination. The keep-only results consequently limit claims about the concentration of useful deltas within full-model checkpoints, while leaving the effectiveness of independent dominant-layer training as a separate empirical observation.

Table 18: Post-hoc keep-only results after full-model MeZO training. Each retained-update model restores all other layers to pretrained weights. DROP reports F1; the other tasks report accuracy. Layer indices accompany retained-update scores.

Model	Task	Zero-shot	Full MeZO	Dominant retained	Best retained
LLaMA2-7B	SST-2	58.02	92.32	L1: 89.33	L1: 89.33
	WSC	36.54	63.46	L1: 38.46	L7: 39.42
	COPA	81.00	86.00	L1: 86.00	L1: 86.00
	DROP	19.73	39.85	L1: 31.15	L1: 31.15
Qwen3-8B	SST-2	58.03	92.11	L6: 94.04	L6: 94.04
	WSC	70.19	70.19	L6: 70.19	L3: 72.12
	COPA	82.00	89.00	L6: 86.00	L6: 86.00
	DROP	63.55	64.12	L6: 63.74	L14: 64.11

D Runtime and Memory Measurements

D.1 Computation Efficiency Analysis

This subsection provides an analytical and empirical breakdown of the runtime consequences of restricting the trainable scope in ZO fine-tuning. The main text argues that dominant-layer and channel-restricted ZO can preserve much of full-model performance while reducing optimization overhead. Here we make that tradeoff explicit by separating the cost of forward computation from the cost of perturbing and updating the selected parameters.

All reported runtime measurements compare ZO methods on LLaMA2-7B using an NVIDIA A6000 GPU (48 GB).

Setup. Consider a decoder-only layer with hidden size H, MLP size I, batch size B, effective sequence length n, and q SPSA directions. Let $F(B, n)$ denote the cost of one full forward pass through the model. For a trainable parameter subset S, let $|S|$ be the number of parameters that are actually perturbed and updated.

Step cost. One MeZO SPSA step performs three parameter perturbations, two full forward passes, and one parameter update. Thus its cost can be written as

$$T_{\text{step}}^{(1)}(S) = 2F(B, n) + \gamma|S|, \tag{3}$$

where $\gamma > 0$ absorbs the per-parameter perturbation and update cost. With q directions,

$$T_{\text{step}}(S) = 2qF(B, n) + q\gamma|S|. \tag{4}$$

Key observation. Equation (4) shows that changing the trainable scope does *not* change the dominant forward computation: full-model, single-layer, and outlier-only MeZO all still require two full forward passes per direction. The only difference lies in the perturbation/update term $q\gamma|S|$.

Parameter counts. For three scopes considered in this paper,

$$|S_{\text{full}}| = P_{\text{full}}, \tag{5}$$

$$|S_{\text{layer}}| = P_{\text{layer}} = 4H^2 + 3HI + 2H, \tag{6}$$

and for k selected MLP outlier channels in one layer,

$$|S_{\text{out}}| = P_{\text{out}} = 3Hk. \tag{7}$$

Therefore,

$$T_{\text{full}} = 2qF + q\gamma P_{\text{full}}, \tag{8}$$

$$T_{\text{layer}} = 2qF + q\gamma P_{\text{layer}}, \tag{9}$$

$$T_{\text{out}} = 2qF + q\gamma P_{\text{out}}. \tag{10}$$

Runtime implication. Assuming perturbation and update are implemented only on the selected parameters, the step-time speedups satisfy

$$\frac{T_{\text{full}}}{T_{\text{layer}}} = \frac{2F + \gamma P_{\text{full}}}{2F + \gamma P_{\text{layer}}}, \qquad \frac{T_{\text{layer}}}{T_{\text{out}}} = \frac{2F + \gamma P_{\text{layer}}}{2F + \gamma P_{\text{out}}}. \tag{11}$$

Hence the achievable end-to-end speedup is always smaller than the raw parameter-count ratio, and approaches that ratio only when perturbation/update dominates the forward cost.

This explains why reducing trainable scope can greatly decrease the perturbation/update overhead, yet may yield only moderate wall-clock speedup when full-sequence forward passes already dominate total runtime.

Table 19: Per-step runtime breakdown for SST2, CB, WSC, COPA, and DROP under fine-tuning different model parameter ranges on LLaMA2-7B. Sparse-MeZO excludes dynamic mask-construction time.

Task	Param Range	Forward/step	Perturb/step	Update/step	Total/step
SST2	full model	0.4431s (39.07%)	0.4849s (42.76%)	0.2061s (18.18%)	1.1341s
SST2	single layer	0.4411s (95.273%)	0.0155s (3.355%)	0.0063s (1.372%)	0.4629s
SST2	1% mlp channel	0.4421s (99.377%)	0.0024s (0.537%)	0.0004s (0.086%)	0.4449s
SST2	Sparse-MeZO	0.4454s (46.96%)	0.3714s (39.16%)	0.1317s (13.89%)	0.9485s
CB	full model	2.9190s (80.82%)	0.4867s (13.48%)	0.2061s (5.7%)	3.6118s
CB	single layer	2.8815s (99.17%)	0.0178s (0.61%)	0.0065s (0.22%)	2.9057s
CB	1% mlp channel	2.8811s (99.68%)	0.0088s (0.3%)	0.0004s (0.02%)	2.8903s
CB	Sparse-MeZO	2.8469s (84.54%)	0.3842s (11.41%)	0.1365s (4.05%)	3.3676s
WSC	full model	1.0724s (60.71%)	0.4858s (27.50%)	0.2083s (11.79%)	1.7665s
WSC	single layer	1.0766s (97.94%)	0.0161s (1.46%)	0.0066s (0.60%)	1.0992s
WSC	1% mlp channel	1.0732s (99.55%)	0.0044s (0.41%)	0.0004s (0.04%)	1.0781s
WSC	Sparse-MeZO	1.0623s (67.67%)	0.3775s (24.05%)	0.1301s (8.29%)	1.5699s
COPA	full model	0.1764s (20.27%)	0.4856s (55.81%)	0.2081s (23.91%)	0.8701s
COPA	single layer	0.1696s (88.17%)	0.0163s (8.49%)	0.0064s (3.34%)	0.1923s
COPA	1% mlp channel	0.1691s (96.70%)	0.0053s (3.05%)	0.0004s (0.25%)	0.1749s
COPA	Sparse-MeZO	0.1655s (25.91%)	0.3490s (54.64%)	0.1242s (19.45%)	0.6387s
DROP	full model	5.8392s (89.38%)	0.4852s (7.43%)	0.2084s (3.19%)	6.5328s
DROP	single layer	5.8273s (99.62%)	0.0158s (0.27%)	0.0066s (0.11%)	5.8496s
DROP	1% mlp channel	5.8334s (99.94%)	0.0029s (0.05%)	0.0004s (0.01%)	5.8368s
DROP	Sparse-MeZO	5.9472s (92.19%)	0.3707s (5.75%)	0.1334s (2.07%)	6.4512s

Table 19 grounds the analytical cost model in measured step-time statistics. Across all tasks, restricting the trainable scope leaves the forward-pass cost nearly unchanged, since the full model must still be executed to obtain the loss. The major difference lies in perturbation and update overhead, which shrinks dramatically for single-layer and 1% MLP-channel settings. Sparse-MeZO falls between these extremes: it reduces perturbation overhead relative to full-model MeZO, but still incurs substantially more optimization-side cost than dominant-layer or channel-restricted tuning. These measurements clarify why scope restriction can deliver substantial end-to-end savings while still falling short of the raw parameter-count reduction when forward evaluation dominates runtime.

On COPA, perturbation and update account for about 80% of a full-model MeZO step, explaining the substantial end-to-end speedup from restricting these operations to one layer. On DROP, forward computation accounts for about 89.4% of the step, leaving less scope for parameter-side savings.

D.2 GPU Memory and Parameter Storage

Peak GPU memory. Table 20 separates the number of trainable parameters, optimizer-state storage, and peak GPU memory. Restricting MeZO to the selected layer reduces the optimized parameter count from 6,738.416M to 202.383M for LLaMA2-7B, and from 8,190.735M to 192.946M for Qwen3-8B. However, the full model remains resident and participates in each forward evaluation, so peak memory changes only from 12.920 to 12.828 GiB and from 16.481 to 16.013 GiB, respectively.

Both MeZO scopes have zero reported optimizer-state storage, while FO AdamW uses 25.103 GiB on LLaMA2-7B and 30.513 GiB on Qwen3-8B. Its measured peak memory is 50.879 and 63.561 GiB, respectively. These results distinguish the forward-only memory benefit relative to FO from the relatively small peak-memory difference between the two MeZO scopes.

Table 20: Parameter counts and GPU memory measurements. Trainable parameters are reported in millions (M), and optimizer-state and peak GPU memory in GiB.

Model	Method	Trainable (M)	Optimizer state	Peak GPU
LLaMA2-7B	Full FO (AdamW)	6,738.416	25.103	50.879
	Full MeZO	6,738.416	0	12.920
	Layer-1 MeZO	202.383	0	12.828
Qwen3-8B	Full FO (AdamW)	8,190.735	30.513	63.561
	Full MeZO	8,190.735	0	16.481
	Layer-6 MeZO	192.946	0	16.013

LoRA parameter storage. For LoRA-based MeZO on LLaMA2-7B, Table 21 compares adapters distributed throughout the model with adapters restricted to layer 1. The trainable parameter count decreases from 19.988M to 0.625M, and the corresponding storage decreases from 38.125 to 1.191 MiB. These measurements concern the trainable adapters only. The pretrained model weights and full forward computation are still required.

Table 21: LoRA trainable-parameter storage for LLaMA2-7B.

Scope	Trainable parameters (M)	Storage (MiB)
Full-model MeZO-LoRA	19.988	38.125
Layer-1 MeZO-LoRA	0.625	1.191

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction match the paper's empirical findings: ZO fine-tuning exhibits a dominant-layer effect.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6 discusses the performance gap relative to first-order fine-tuning, the number of optimization steps, and extensions to additional model architectures, scales, and ZO optimizers.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The main claims are empirical and are supported by experimental ablations and analyses rather than theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5 and Appendix A specify the models, datasets, data budgets, evaluation protocols, optimization settings, hyperparameter grids, training steps, and random seeds. The public repository linked in the abstract provides implementation and execution instructions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at `https://github.com/uncc-efficient-ai/dominant-layer-zo`. The repository provides dependency specifications, model and method configurations, training and evaluation commands, and instructions for accessing the public models and datasets. Experimental settings are reported in Section 5 and Appendix A.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The main paper and appendix report the training and evaluation details, including datasets and splits, evaluated models, prompt/evaluation protocols, trainable parameter settings, optimizer choices, hyperparameter search ranges, training steps, batch sizes, and evaluation metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 5 reports mean $\pm$ standard deviation across training seeds 0, 42, and 2026 for full-model and dominant-layer MeZO on four tasks for LLaMA2-7B and Qwen3-8B. These statistics describe variation across training seeds; the remaining experiments use seed 0 unless otherwise stated.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix A reports GPU types and numerical precision, and Appendices D and D.2 report LLaMA2-7B ZO runtimes and peak GPU memory measurements, respectively. Per-run execution times for every configuration and total project compute are not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The work only uses publicly available models and benchmark datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [No]

    Justification: The paper studies optimization behavior and computational efficiency rather than a particular deployment. It does not include a dedicated assessment of positive and negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new high-risk models or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper cites the model and dataset sources, and the code repository provides third-party attribution and license notices. The manuscript does not explicitly enumerate the license and terms of use for every model and dataset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The released code repository includes setup and dependency information, training and evaluation instructions, model and method configurations, and third-party notices. No new datasets or pretrained foundation models are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research, user studies, participant data collection, or crowdsourcing. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM assistance was limited to writing, editing, and formatting; it was not an original or non-standard component of the research methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.