RETHINKING LOCAL LEARNING: A CHEAPER AND FASTER RECIPE FOR LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM post-training typically propagates task gradients through the full depth of the model, coupling adaptation to full-depth activation storage and direct task-gradient access to pretrained representations. We study whether this backward coupling is necessary when post-training supervision is narrower than pretraining. We propose LoPT: Local-Learning Post-Training, which places a single gradient boundary at the transformer midpoint. The second-half block follows the task objective, while the first-half block remains trainable through a lightweight feature-reconstruction objective without receiving direct task gradients. This asymmetric objective assignment shortens the task-induced backward path while retaining local adaptability. Across the reported SFT and GRPO settings, LoPT achieves competitive quality with lower peak memory and improved profiled training efficiency, and reduces held-out capability degradation in several SFT settings. A separate three-seed assessment against an LR-selected E2E baseline further characterizes the trade-off between validation fitting and benchmark performance. The results support task-gradient reach as a practical design choice alongside objectives, parameterization, and execution schedules.

1 INTRODUCTION

Post-training methods such as supervised fine-tuning (SFT), direct preference optimization (DPO), and reinforcement-learning variants shape model behavior after pre-training (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024). Despite different objectives, they commonly propagate task gradients through the model's full depth. This simple, general pathway is usually treated as an implementation default rather than a design choice.

Work on objectives, parameter-efficient updates, and systems optimization does not fully characterize this pathway: even when few parameters are updated, the task loss may induce a backward graph spanning the transformer stack. We therefore study *gradient reach*—how far the task loss propagates—as a variable governing activation storage, backward dependencies, and which pretrained representations receive task gradients.

Post-training supervision is often much narrower than pre-training, emphasizing one format, task family, or reward. Yet end-to-end training still couples adaptation to full-depth activation storage, long-range credit assignment, and direct task-gradient updates to broadly pretrained representations. We call this *full-depth backward coupling*. This coupling can be costly when adaptation is concentrated in task-facing layers and intrusive when narrow gradients perturb general lower-layer features.

Supervised local-learning methods shorten backpropagation using local auxiliary objectives (Su et al., 2024a; Zhang et al., 2025; Su et al., 2026). We introduce **Local-Learning Post-Training (LoPT)**, which partitions the transformer into first- and second-half blocks k_1 and k_2 with a midpoint `detach()` boundary. The task objective updates k_2, while a lightweight reconstruction objective keeps k_1 trainable without direct task gradients. The local objective constrains lower-block adaptation rather than solving the downstream task independently. LoPT therefore controls gradient reach rather than the number of trainable parameters and is orthogonal to LoRA-style parameterization.

By shortening the task-induced backward path while retaining a local update below the boundary, LoPT separates task-facing adaptation from lower-block maintenance. It also composes with LoRA, gradient checkpointing, DeepSpeed/ZeRO, and pipeline parallelism.

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

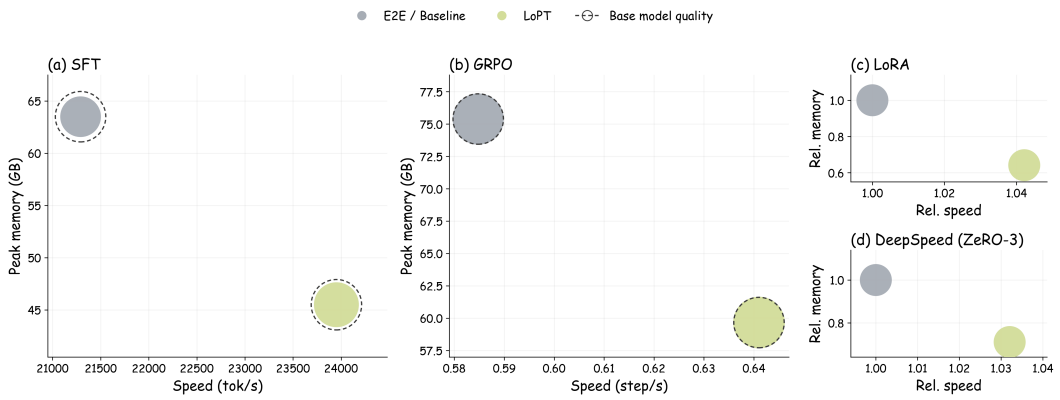


Figure 1: **LoPT quality–efficiency comparisons in the original study.** (a) Qwen2.5-7B-Instruct on Tulu-3 SFT, using the measurements in Table 7. (b) Qwen2.5-7B-Instruct on GSM8K GRPO; the horizontal axis is policy-update steps per second, the reciprocal of update time in Table 2. Bubble size represents the seven-benchmark mean, and dashed rings indicate base-model quality. Panels (c,d) show the matched-stack LoRA and DeepSpeed-ZeRO3 profiles in Table 18.

Our main contributions are summarized as follows.

- We formulate task-gradient reach as a post-training design variable connecting memory, backward dependencies, and direct task-gradient access to pretrained representations.
- We propose **LoPT**, which uses one midpoint boundary, a task objective for the second half, and lightweight feature reconstruction for the first half.
- Across SFT and GRPO settings, LoPT provides competitive quality while reducing training memory and often improving efficiency.

We discuss related work on local learning and memory-efficient LLM post-training in Appendix B.

2 METHOD

We formalize end-to-end post-training, introduce LoPT's stop-gradient boundary, and instantiate the method for supervised fine-tuning (SFT) and Group Relative Policy Optimization (GRPO).

2.1 PRELIMINARIES

Consider an autoregressive transformer (Vaswani et al., 2017) with an embedding layer, N transformer layers, a final normalization layer, and an output head. Given an input token sequence $\mathbf{x}$, we denote its embedding sequence by $\mathbf{h}_0 = \texttt{Embed}(\mathbf{x})$. We split the transformer at layer s into two consecutive blocks, a first block k_1 and a second block k_2:

$$\begin{aligned} k_1 &= \{\text{Layer}_0, \ldots, \text{Layer}_{s-1}\}, \\ k_2 &= \{\text{Layer}_s, \ldots, \text{Layer}_{N-1}, \texttt{norm}, \texttt{lm_head}\}. \end{aligned} \tag{1}$$

Let θ_1 include the input-embedding parameters and the first-half transformer parameters, and let θ_2 contain the second-half transformer, final normalization, and output-head parameters. The block notation in Eq. 1 lists the transformer layers; $\mathbf{h}_0 = \text{Embed}(\mathbf{x})$ is produced by the input embedding included in the first parameter group. The forward computation is

$$\mathbf{h}_1 = f_{k_1}(\mathbf{h}_0; \theta_1), \qquad \mathbf{y} = f_{k_2}(\mathbf{h}_1; \theta_2), \tag{2}$$

where $\mathbf{h}_1$ is the intermediate representation passed from the first block to the second block, and $\mathbf{y}$ denotes the output logits.

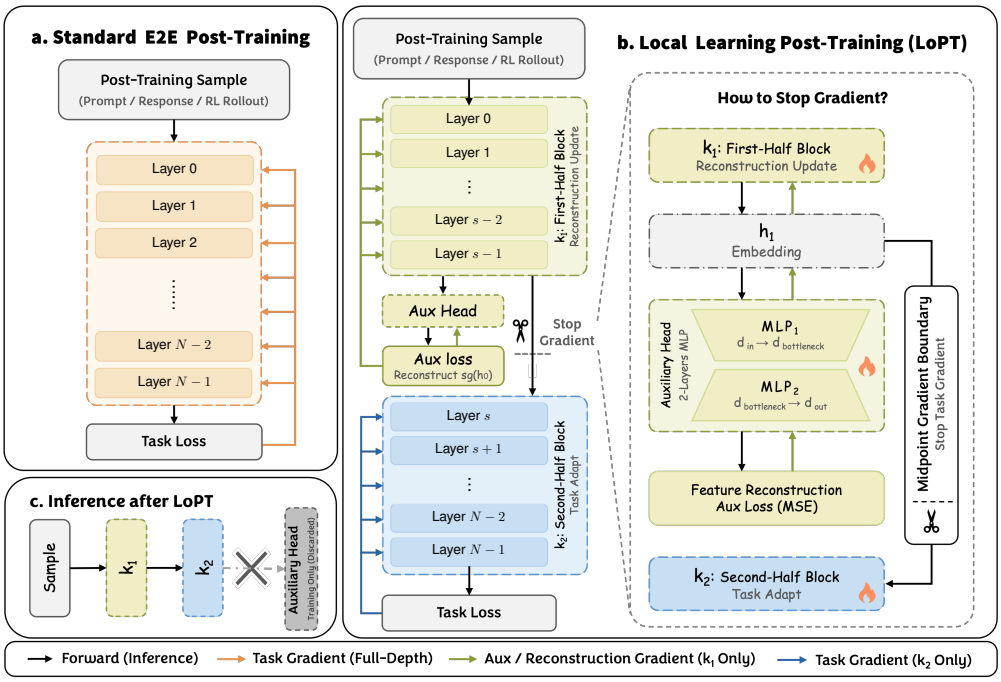


Figure 2: **Overview of Local-Learning Post-Training (LoPT).** (a) Standard E2E post-training propagates task gradients through the full transformer stack. (b) LoPT inserts a midpoint stop-gradient boundary: k_2 is updated by the task loss, while k_1 remains trainable through a local feature-reconstruction objective implemented with a lightweight auxiliary head. The boundary activation produced by k_1 is reused directly by k_2 after detachment; LoPT does not explicitly rerun k_1 after the auxiliary update to refresh this activation. (c) The auxiliary head is used only during training and discarded at inference, so inference follows the standard $k_1 \rightarrow k_2$ forward path without extra computation. Transformer layers are indexed from zero, and $s = \lfloor N/2 \rfloor$ marks the first layer of the second block.

The update expressions below use gradient-descent notation to specify gradient routing. The experiments use AdamW-family optimizers with parameter-group-specific learning rates and optimizer states.

Under standard end-to-end post-training, the above split is only notational. The task loss $\mathcal{L}_{\text{task}}$ is backpropagated through the full model by:

$$\begin{aligned} \theta_1 &\leftarrow \theta_1 - \eta_1 \nabla_{\theta_1} \mathcal{L}_{\text{task}}, \quad \nabla_{\theta_1} \mathcal{L}_{\text{task}} = \frac{\partial \mathcal{L}_{\text{task}}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1}{\partial \theta_1}, \\ \theta_2 &\leftarrow \theta_2 - \eta_2 \nabla_{\theta_2} \mathcal{L}_{\text{task}}, \quad \nabla_{\theta_2} \mathcal{L}_{\text{task}} = \frac{\partial \mathcal{L}_{\text{task}}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \theta_2}. \end{aligned} \tag{3}$$

2.2 Local-Learning View of Gradient Reach

LoPT restricts task supervision to the second-half block. The first-half block remains on the forward path but receives no direct task gradient (Fig. 2).

Concretely, LoPT inserts a stop-gradient boundary between k_1 and k_2 by:

$$\mathbf{h}_1 = f_{k_1}(\mathbf{h}_0; \theta_1), \qquad \hat{\mathbf{h}}_1 = \text{sg}(\mathbf{h}_1), \qquad \mathbf{y} = f_{k_2}(\hat{\mathbf{h}}_1; \theta_2), \tag{4}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator, acting as the identity in the forward pass while setting the backward Jacobian to zero.

The task loss therefore remains a function of the representation produced by the first-half block, while its gradient is prevented from reaching the first-half-block parameters:

$$\begin{aligned}\nabla_{\theta_1}\mathcal{L}_{\text{task}} &= \frac{\partial \mathcal{L}_{\text{task}}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \hat{\mathbf{h}}_1}\underbrace{\frac{\partial \hat{\mathbf{h}}_1}{\partial \mathbf{h}_1}}_{=\mathbf{0}}\frac{\partial \mathbf{h}_1}{\partial \theta_1} = \mathbf{0},\\ \nabla_{\theta_2}\mathcal{L}_{\text{task}} &= \frac{\partial \mathcal{L}_{\text{task}}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \theta_2}.\end{aligned} \tag{5}$$

Thus, the task-induced backward path terminates at the boundary. Since the boundary alone would make the first-half block passive with respect to training, LoPT assigns it a local auxiliary objective:

$$\mathcal{L}_{\text{local}} = \mathcal{L}_{\text{local}}(\mathbf{h}_0, \mathbf{h}_1; \phi), \tag{6}$$

where ϕ denotes the auxiliary parameters. This feature-level objective keeps the first-half block trainable without direct task gradients. Reconstruction encourages recovery of input information, not equality with pretrained representations. The updates are

$$\begin{aligned}\theta_1 &\leftarrow \theta_1 - \eta_1 \nabla_{\theta_1}\big(\lambda_{\text{local}}\mathcal{L}_{\text{local}}\big),\\ \phi &\leftarrow \phi - \eta_\phi \nabla_{\phi}\big(\lambda_{\text{local}}\mathcal{L}_{\text{local}}\big),\\ \theta_2 &\leftarrow \theta_2 - \eta_2 \nabla_{\theta_2}\mathcal{L}_{\text{task}}.\end{aligned} \tag{7}$$

Here η_1, η_ϕ, and η_2 denote the learning rates of the first-half, auxiliary-head, and second-half parameter groups. They need not be equal. Thus, blocking task gradients does not require freezing the first-half block.

2.3 Local Learning for LLM Post-Training

LoPT controls the reach of task supervision by changing the backward path while leaving the post-training objective unchanged. To define the gradient boundary, we use the same partition as in Eq. 1 and set

$$s = \left\lfloor \frac{N}{2} \right\rfloor. \tag{8}$$

After this choice, k_1 and k_2 correspond to the first-half and second-half blocks, respectively. The midpoint boundary is used as a simple architecture-agnostic default, avoiding split-location tuning while making gradient reach comparable across model families. For architectures with tied input embeddings and output heads, we treat the two modules as distinct parameter tensors when defining the split, preventing output-head gradients from bypassing the LoPT boundary and updating the input embedding.

Training objectives of LoPT. LoPT keeps the standard supervised post-training objective unchanged in form, but changes how its gradients are routed through the model. The task branch receives the boundary representation produced by the first-half block while stopping the task gradient at the boundary. Formally, we compute

$$\mathbf{h}_1 = f_{k_1}(\mathbf{h}_0), \qquad \hat{\mathbf{h}}_1 = \text{sg}(\mathbf{h}_1). \tag{9}$$

For supervised fine-tuning, we use the standard causal language-modeling loss over the non-padding prediction positions in the formatted instruction-output sequence:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \log p_{\theta_2}\left(x_{t+1} \mid x_{\le t};\, \hat{\mathbf{h}}_1\right), \tag{10}$$

where $\mathcal{T}$ denotes the set of non-padding prediction positions used for the SFT loss. The forward prediction still depends on both halves of the model because $\hat{\mathbf{h}}_1$ is produced by k_1, but for the SFT objective it is treated as a constant in backpropagation:

$$\nabla_{\theta_1}\mathcal{L}_{\text{SFT}} = \mathbf{0}, \qquad \nabla_{\theta_2}\mathcal{L}_{\text{SFT}} \not\equiv \mathbf{0}. \tag{11}$$

For GRPO (Shao et al., 2024), given a prompt $\mathbf{x}$, sampled responses $\{o_i\}_{i=1}^{G}$, and normalized advantages $\hat{A}_i$, the current policy log-probability is computed through the same stop-gradient boundary:

$$\begin{aligned} \mathbf{h}_{1,i,t} &= f_{k_1}(\mathbf{h}_{0,i,t}; \theta_1), \\ \hat{\mathbf{h}_{1,i,t}} &= \mathrm{sg}(\mathbf{h}_{1,i,t}), \\ \ell_{i,t}^{\mathrm{LoPT}} &= \log \pi_{\theta_2}\left(o_{i,t} \mid \mathbf{x}, o_{i,<t}; \hat{\mathbf{h}_{1,i,t}}\right). \end{aligned} \tag{12}$$

The token-level probability ratio and clipped GRPO loss are computed as:

$$\begin{aligned} r_{i,t} &= \exp\left(\ell_{i,t}^{\mathrm{LoPT}} - \log \pi_{\mathrm{old}}\left(o_{i,t} \mid \mathbf{x}, o_{i,<t}\right)\right), \\ \mathcal{L}_{\mathrm{GRPO}} &= -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left(r_{i,t}\hat{A}_i,\ \mathrm{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\right). \end{aligned} \tag{13}$$

Although the current policy is evaluated using the boundary representation produced by k_1, the stop-gradient boundary prevents the resulting policy-gradient signal from entering the first-half block:

$$\nabla_{\theta_1}\mathcal{L}_{\mathrm{GRPO}} = \mathbf{0}, \qquad \nabla_{\theta_2}\mathcal{L}_{\mathrm{GRPO}} \not\equiv \mathbf{0}. \tag{14}$$

Feature reconstruction for local adaptability. A lightweight bottleneck MLP reconstructs input embeddings from the boundary representation, supplying the first-half block with a local training signal:

$$g_\phi(\mathbf{h}) = W_2\,\mathrm{GELU}(W_1\,\mathrm{LayerNorm}(\mathbf{h}) + \mathbf{b}_1) + \mathbf{b}_2, \tag{15}$$

where $W_1 \in \mathbb{R}^{\frac{d}{4}\times d}$ and $W_2 \in \mathbb{R}^{d\times\frac{d}{4}}$. The auxiliary objective then can be written as:

$$\mathcal{L}_{\mathrm{aux}} = \frac{1}{BLd}\sum_{b=1}^{B}\sum_{\ell=1}^{L}\|g_\phi(\mathbf{h}_{1,b,\ell}) - \mathrm{sg}(\mathbf{h}_{0,b,\ell})\|_2^2. \tag{16}$$

Here B is the batch size, L is the represented sequence length, and d is the hidden dimension.

Reconstruction updates the first-half block and auxiliary head; the task loss updates only the second-half block. Preservation of pretrained capabilities is assessed by the benchmarks, not guaranteed by reconstruction. The updates are

$$\begin{aligned} \theta_1 &\leftarrow \theta_1 - \eta_1\nabla_{\theta_1}\left(\lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}}\right), \\ \phi &\leftarrow \phi - \eta_\phi\nabla_{\phi}\left(\lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}}\right), \\ \theta_2 &\leftarrow \theta_2 - \eta_2\nabla_{\theta_2}\mathcal{L}_{\mathrm{task}}, \qquad \mathcal{L}_{\mathrm{task}} \in \{\mathcal{L}_{\mathrm{SFT}}, \mathcal{L}_{\mathrm{GRPO}}\}. \end{aligned} \tag{17}$$

Boundary reuse and update scheduling. LoPT passes the detached first-half activation to k_2 without explicitly recomputing it after the local update. The auxiliary backward is completed before constructing the corresponding task graph, allowing the first-half graph to be released earlier. Microbatch backward operations are distinct from optimizer steps: with gradient accumulation, parameter updates and scheduler advances occur at update-window boundaries. Checkpoint-induced activation replay is separate from refreshing the boundary supplied to the task branch. Appendix J describes the execution order, and Appendix O compares the profiled schedules. The auxiliary head is removed at inference.

Implementation checks. Tests against schedule-specific references match the reported losses, gradient norms, parameter updates, optimizer-state transitions, and update counts. They also record non-zero changes in the tested lower components and agreement with a frozen reference when their learning rate is zero (Appendix P).

3 Experiments

3.1 Implementations

3.1.1 Models and Datasets

We train LoPT on Qwen3-4B-Base (Yang et al., 2025), Qwen2.5-7B-Instruct (Yang et al., 2024), and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). For supervised fine-tuning, we use Alpaca-52K (Taori et al., 2023), Tulu-3 SFT Mix (Lambert et al., 2024), Magpie-Pro (Xu et al., 2024),

and MetaMathQA (Yu et al., 2023), covering both general instruction tuning and math-focused supervised adaptation. For GRPO, we use GSM8K-train (Cobbe et al., 2021) and NuminaMath (Li et al., 2024) as training datasets.

Experimental protocol. We evaluate LoPT under both supervised fine-tuning (SFT) and GRPO. For SFT, we use Alpaca-52K and MetaMathQA as the main training datasets and report results on Qwen3-4B-Base, Qwen2.5-7B-Instruct, and Llama-3.1-8B-Instruct. For GRPO, we train on GSM8K and evaluate the resulting policies under the same benchmark suite. Unless otherwise stated, models are evaluated with lm-eval-harness on seven benchmarks covering instruction following, reasoning, factuality, commonsense, and language understanding. Full training hyperparameters, decoding settings, and evaluation details are provided in Appendix C.1 and Appendix C.2.

3.2 Main Results

3.2.1 Quality–Efficiency Trade-off

Tables 1 and 2 compare LoPT with end-to-end post-training in quality and training cost. Base denotes the pretrained checkpoint, and E2E denotes full-depth task-gradient propagation. SFT results cover Alpaca-52K and MetaMathQA. The GSM8K GRPO comparison uses the same rollout procedure and reports policy-update cost separately; these timings exclude rollout generation, reward evaluation, and synchronization outside the update phase.

SFT. On Alpaca-52K, E2E degrades several non-rehearsed benchmarks: for Llama-3.1-8B-Instruct, GSM8K and IFEval fall from 69.67/44.54 to 46.62/30.50, while LoPT reaches 72.10/46.03. Average LoPT-over-E2E gains are $+1.32$, $+1.97$, and $+7.22$ across the three model blocks. On the more aligned MetaMathQA data, LoPT is mostly near parity, with a $+2.77$ average gain for Llama-3.1-8B-Instruct. In the original matched SFT runs of Table 1, LoPT reduces peak memory by approximately 24–36% and increases throughput by approximately 2–7%.

GRPO. Under GRPO, GSM8K gains remain below 0.4 pp, while held-out effects are modest and mixed. In Table 2, LoPT reduces policy-update peak memory by 21–24% and update time by 7–21%. Its midpoint boundary reduces task-branch activation lifetime and task-backward work below it, although total work includes the auxiliary k_1 update. Appendix F reports compatibility with checkpointing, LoRA, DeepSpeed-ZeRO, and pipeline parallelism.

Table 1: SFT evaluation results and training cost on lm-eval-harness. Each model is trained on Alpaca-52K or MetaMathQA and evaluated under Base, E2E, and LoPT. Peak memory is measured in GB, and speed is measured as total training throughput in tokens/s.

Dataset	Model	Method	MMLU↑	IFEval↑	ARC-C↑	GSM8K↑	HellaSwag↑	TruthfulQA↑	Winogrande↑	Peak Mem (GB)↓	Speed (tok/s)↑
Alpaca-52K	Qwen3-4B-Base	Base	70.07	23.84	57.84	**85.14**	65.62	**54.85**	67.56	-	-
		E2E	**70.12**	25.87	61.77	79.45	69.80	53.07	**68.19**	31.5	21,020
		LoPT (ours)	69.99	**29.39**	**61.95**	84.62	**69.97**	53.91	67.69	21.9	21,469
	Qwen2.5-7B-Ins	Base	**74.17**	**57.30**	**66.72**	**83.47**	79.35	**64.75**	74.51	-	-
		E2E	73.09	40.30	60.58	74.22	78.55	52.28	**75.77**	57.7	15,561
		LoPT (ours)	74.08	41.22	63.96	79.38	**79.46**	54.86	75.63	37.2	15,971
	Llama-3.1-8B-Ins	Base	68.20	44.54	**60.49**	69.67	**79.42**	**54.56**	**77.50**	-	-
		E2E	**68.67**	30.50	57.59	46.62	78.31	47.39	76.09	60.6	14,483
		LoPT (ours)	68.48	**46.03**	60.41	**72.10**	79.38	52.67	76.64	38.9	14,882
MetaMathQA	Qwen3-4B-Base	Base	70.07	**23.84**	57.84	**85.14**	65.62	**54.85**	67.56	-	-
		E2E	**70.18**	20.89	63.57	81.20	72.43	54.49	68.69	33.0	25,019
		LoPT (ours)	70.17	22.26	**63.64**	81.21	**72.53**	54.55	**68.88**	25.2	26,670
	Qwen2.5-7B-Ins	Base	74.17	**57.30**	**66.72**	83.47	79.35	**64.75**	**74.51**	-	-
		E2E	74.13	51.76	63.91	86.80	80.85	62.63	71.51	58.7	19,567
		LoPT (ours)	**74.23**	51.46	64.33	**87.21**	**80.97**	62.66	74.03	38.4	20,511
	Llama-3.1-8B-Ins	Base	**68.20**	**44.54**	60.49	69.67	79.42	**54.56**	**77.50**	-	-
		E2E	65.29	34.94	58.70	76.79	77.70	52.06	76.56	61.4	18,028
		LoPT (ours)	67.94	43.99	**60.92**	**79.27**	**79.71**	52.93	76.64	39.9	19,194

Table 2: GRPO evaluation results and policy-update cost on lm-eval-harness. Models are trained on GSM8K and evaluated under Base, E2E, and LoPT. Peak memory is measured in GB, and step time is measured as policy-update time in s/step.

Dataset	Model	Method	MMLU↑	IFEval↑	ARC-C↑	GSM8K↑	HellaSwag↑	TruthfulQA↑	Winogrande↑	Peak Mem (GB)↓	Step Time (s/step)↓
GSM8K	Qwen3-4B-Base	Base	70.07	23.84	57.84	85.14	65.62	**54.85**	**67.56**	-	-
		E2E	70.01	**24.03**	62.03	88.35	68.76	54.78	67.17	42.55	1.42
		LoPT (ours)	**70.10**	23.84	**62.29**	**88.63**	**69.95**	54.62	66.85	32.89	1.32
	Qwen2.5-7B-Ins	Base	74.17	57.30	**66.72**	83.47	79.35	64.75	74.51	-	-
		E2E	**74.22**	56.01	**66.72**	88.70	81.23	64.52	75.22	75.39	1.71
		LoPT (ours)	74.18	**59.70**	66.38	**89.09**	**81.40**	**64.83**	**75.40**	59.66	1.56
	Llama-3.1-8B-Ins	Base	68.20	44.54	60.49	69.67	79.42	**54.56**	**77.50**	-	-
		E2E	68.44	**47.69**	**61.69**	82.02	80.39	54.53	76.80	62.62	2.10
		LoPT (ours)	**68.55**	47.03	61.43	**82.40**	**80.47**	54.11	77.35	47.54	1.65

Three-seed comparison. We also compare the three methods on Qwen2.5-7B-Instruct and MetaMathQA using training seeds 1001–1003. E2E's learning rate is selected as 10^{-5} by validation NLL on HPO seed 11. LoPT's seven-task mean is higher for all three paired seeds, with an average difference of 2.952 ± 0.314 percentage points; the uncertainty is the sample standard deviation of the paired differences. Task-level gains occur on GSM8K-CoT and WinoGrande, whereas E2E has lower validation NLL. Appendix N gives the full results and the training and evaluation settings, which differ from those of the main study.

Execution-order diagnostic. Under a common profiling workload, cached-boundary local execution uses less allocated memory than joint execution. Compared with post-update recomputation, it uses 0.200 GiB more allocated memory but has 12.09% higher measured input-token throughput. This compares the cost of execution schedules; the complete measurements and their scope are reported in Appendix O.

3.2.2 Localized Parameter Drift

LoPT keeps k_1 on the forward path and trainable through reconstruction, while the task objective updates k_2. We examine where parameter movement is concentrated in the measured checkpoints; gradient isolation alone does not impose a bound on the magnitude of auxiliary-driven updates.

We measure layer-wise relative parameter drift between the tuned and base checkpoints:

$$\Delta_\ell = \frac{\left\|\theta_\ell^{\text{tuned}} - \theta_\ell^{\text{base}}\right\|_2}{\left\|\theta_\ell^{\text{base}}\right\|_2}. \tag{18}$$

In the profiled GRPO checkpoints, Figure 3 shows that E2E distributes relative parameter change across both blocks, whereas LoPT keeps k_1 closer to the base checkpoint and concentrates more movement in k_2. The attention/MLP decomposition exhibits the same empirical pattern.

Table 3(a) provides a separate sensitivity check by perturbing the first 14 base-model layers at relative scale 2×10^{-7}. The seven-task mean changes from 71.47 to 70.93 (-0.54 pp). This diagnostic shows that small early-layer perturbations can coincide with measurable score changes; it is neither a matched post-training control nor evidence that random movement is beneficial. The implementation checks in Appendix P, rather than this perturbation experiment, establish non-zero local updates in the tested LoPT cases.

3.3 Ablation Study

3.3.1 Reconstruction-Guided Updates versus Freezing the First-Half Block.

In the original matched settings, freeze-k_1 removes LoPT's first-half reconstruction loss and update. LoPT improves the seven-task mean by 1.24 pp in SFT and 1.64 pp in GRPO (Table 4), providing evidence for a benefit of the local update in these two settings. Table 3(a) is a separate base-model sensitivity check and does not establish the cause of this improvement.

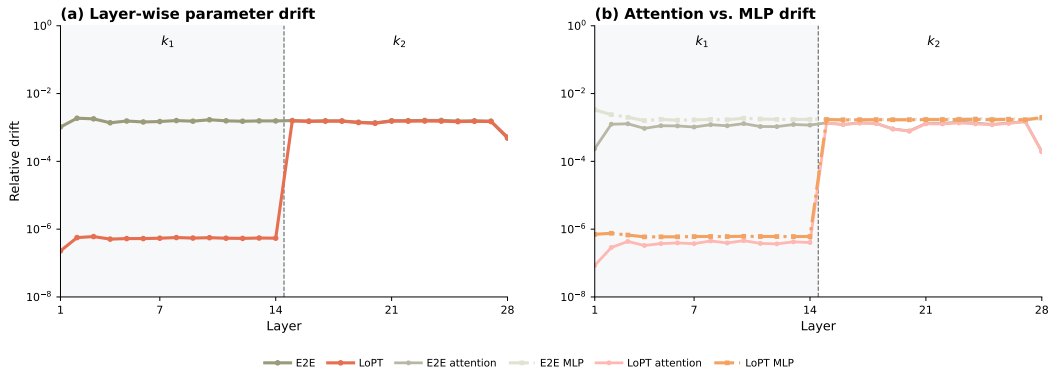


Figure 3: **Layer-wise parameter drift under GRPO.** Drift is measured as the relative parameter change from the base checkpoint to the final tuned checkpoint; the axis uses one-based layer numbers. In the measured checkpoints, LoPT concentrates persistent movement in k_2, whereas E2E spreads movement across the full model.

Table 3: **Mechanism diagnostics and a separate configuration-level assessment.** **(a)** Front-layer perturbation sensitivity of the Qwen2.5-7B-Instruct base checkpoint under the original evaluation protocol; scores are percentages. **(b)** Selected-validation answer-token NLL and seven-task mean for the additional three-seed assessment; entries are mean ± sample standard deviation. Macro scores are percentages and their standard deviations are percentage points. The panels use different protocols and are not a matched intervention pair.

(a) Front-layer perturbation

Benchmark	Base	Perturbed	Δ
MMLU	**74.17**	73.26	-0.91
IFEval	57.30	**58.59**	+1.29
ARC-C	**66.72**	65.70	-1.02
GSM8K	**83.47**	82.01	-1.46
HellaSwag	**79.35**	78.87	-0.48
TruthfulQA	**64.75**	64.02	-0.73
Winogrande	**74.51**	74.03	-0.48
Avg.	**71.47**	70.93	-0.54

(b) Additional assessment

Method	NLL↓	Macro (%)↑
E2E	**0.137346** ±0.000038	60.7501 ±0.1807
freeze-k_1	0.141267 ±0.000218	**64.0075** ±0.2547
LoPT	0.139754 ±0.000111	63.7021 ±0.2710

Configurations differ in optimization settings; this is not a reconstruction-only ablation.

Panel (a) perturbs only the first 14 layers at relative scale 2×10^{-7}. Panel (b) uses the separately specified protocol in Appendix N, including TruthfulQA-MC1, GSM8K-CoT 8-shot, and HellaSwag 10-shot. Optimization settings also differ across the three methods.

Fitting and evaluation relative to freezing. In Table 3(b), LoPT has lower selected-validation NLL than freezing (0.139754 versus 0.141267), but a slightly lower seven-task mean (63.7021% versus 64.0075%). E2E has the lowest NLL, 0.137346, and a seven-task mean of 60.7501%. Learning rates, AdamW β_2, and warmup differ across these configurations (Table 33), so the comparison does not isolate the reconstruction update. Table 4 reports the separate matched controls from the original study.

3.3.2 Ablation on the Number of Partitions

Table 5 compares the default two-block design ($k = 2$) with a four-block variant.

The four-block variant improves IFEval but is lower on the other six benchmarks, including GSM8K. This comparison supports retaining the simpler two-block configuration in the tested setting. It does not isolate the effect of moving a single boundary, because partition granularity also changes; midpoint placement is therefore a simple default rather than an empirically established universal optimum.

Table 4: **freeze-k_1 ablations in SFT and GRPO.** Values are percentages. Δ denotes LoPT (ours) minus freeze-k_1.

	SFT: MetaMathQA / Qwen2.5-7B-Ins					GRPO: GSM8K / Qwen2.5-7B-Ins				
Benchmark	Base	E2E	freeze-k_1	LoPT (ours)	Δ	Base	E2E	freeze-k_1	LoPT (ours)	Δ
MMLU	74.17	74.13	74.06	**74.23**	+0.17	74.17	74.22	**74.20**	74.18	-0.02
IFEval	57.30	51.76	50.46	**51.46**	**+1.00**	57.30	56.01	57.96	**59.70**	**+1.74**
ARC-C	66.72	63.91	63.76	**64.33**	+0.57	66.72	66.72	64.55	**66.38**	**+1.83**
GSM8K	83.47	86.80	85.91	**87.21**	**+1.30**	83.47	88.70	87.56	**89.09**	**+1.53**
HellaSwag	79.35	80.85	78.84	**80.97**	**+2.13**	79.35	81.23	79.37	**81.40**	**+2.03**
TruthfulQA	64.75	62.63	60.61	**62.66**	**+2.05**	64.75	64.52	62.72	**64.83**	**+2.11**
Winogrande	74.51	71.51	72.56	**74.03**	**+1.47**	74.51	75.22	73.16	**75.40**	**+2.24**
Avg. Δ					**+1.24**					**+1.64**

Bolding compares freeze-k_1 and LoPT; Base and E2E provide context. Green marks $\Delta \geq +1$ pp, gray marks $|\Delta| < 1$ pp, and the comparison is limited to these diagnostic settings.

Table 5: **Number-of-partitions ablation on Qwen2.5-7B-Instruct under the original evaluation protocol.** Here k denotes the number of blocks, not the split-layer index. All reported scores are percentages.

Method	MMLU	IFEval	ARC-C	GSM8K	HellaSwag	TruthfulQA	Winogrande
E2E	74.13	51.76	63.91	86.80	80.85	62.63	71.51
LoPT $k = 2$	**74.23**	51.46	**64.33**	**87.21**	**80.97**	**62.66**	**74.03**
LoPT $k = 4$	74.09	**52.60**	63.18	85.59	80.30	62.49	73.09

4 Conclusion

We presented LoPT, a local-learning strategy that makes task-gradient reach an explicit choice in LLM post-training. A midpoint boundary separates second-half task optimization from first-half feature reconstruction. Unlike rigid freezing, the first half remains trainable under a local objective, and the auxiliary head is removed for inference.

Across the reported SFT and GRPO experiments, LoPT achieves competitive quality with lower training or policy-update memory and improved profiled efficiency. Its strongest quality gains occur in selected SFT settings where E2E degrades held-out benchmarks; the GRPO results primarily support lower-memory policy updates at near-parity quality. The execution-order diagnostic also shows the practical value of local backpropagation with a cached boundary.

The additional three-seed assessment finds a positive benchmark-average gap over an LR-selected E2E baseline, alongside a nonuniform fitting–evaluation trade-off relative to E2E and freezing. Together, the results support task-gradient reach as a practical design dimension while showing that LoPT's empirical benefits remain dependent on the training and evaluation setting.

Limitations. The original experiments are predominantly single-run, and the additional three-seed study covers one setting. Split-location sensitivity, stronger preservation baselines, and direct semantic-preservation evidence remain unestablished. GRPO timings exclude rollout and reward computation, and the 32B result is only a sanity check.

AI USE STATEMENT

Generative AI tools were used to assist with experimental planning, implementation and debugging support, analysis of experimental results, manuscript editing, and LaTeX formatting. The authors are responsible for the final manuscript and all reported claims, results, and artifacts.

ETHICS STATEMENT

This work studies post-training methods using publicly released models and datasets; it does not involve human-subject data collection or deployment of a user-facing system. Lowering post-training cost may broaden access to model adaptation, but it may also lower barriers to misuse of the underlying models. The method does not introduce a new application-specific risk beyond those associated with the models and datasets being adapted.

REPRODUCIBILITY STATEMENT

Section 2 specifies the partition, objectives, and gradient routing. Appendices C–M describe the original protocols, profiling configurations, metrics, compute, and auxiliary architecture. Appendices N–P report the separate three-seed assessment, execution-order profiles, and implementation checks. The original study is single-run per configuration unless otherwise stated; three-seed uncertainty applies only to the additional assessment. Unreported configuration fields and the scope of the measurements are identified in the corresponding appendices.

REFERENCES

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pp. 736–745. PMLR, 2020.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

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

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

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*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pp. 1627–1635. PMLR, 2017.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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)*, pp. 4582–4597, 2021.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Qijun Luo, Hengxu Yu, and Xiao Li. BAdam: A memory efficient full parameter optimization method for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2404.02827.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pp. 4839–4850. PMLR, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Junhao Su, Changpeng Cai, Feiyu Zhu, Chenghao He, Xiaojie Xu, Dongzhi Guan, and Chenyang Si. Momentum auxiliary network for supervised local learning. In *European Conference on Computer Vision*, pp. 276–292. Springer, 2024a.

Junhao Su, Chenghao He, Feiyu Zhu, Xiaojie Xu, Dongzhi Guan, and Chenyang Si. Hpff: Hierarchical locally supervised learning with patch feature fusion. In *European Conference on Computer Vision*, pp. 293–309. Springer, 2024b.

Junhao Su, Feiyu Zhu, Hengyu Shi, Tianyang Han, Yurui Qiu, Junfeng Luo, Xiaoming Wei, and Jialin Gao. Man++: Scaling momentum auxiliary network for supervised local learning in vision tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: an alternative to end-to-end training. *arXiv preprint arXiv:2101.10832*, 2021.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024. URL `https://api.semanticscholar.org/CorpusID:271212307`.

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.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

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

Yuming Zhang, Shouxin Zhang, Peizhe Wang, Feiyu Zhu, Dongzhi Guan, Junhao Su, Jiabin Liu, and Changpeng Cai. Mlaan: Scaling supervised local learning with multilaminar leap augmented auxiliary network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22686–22694, 2025.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Feiyu Zhu, Yuming Zhang, Xiuyuan Guo, Hengyu Shi, Junfeng Luo, Junhao Su, and Jialin Gao. Advancing supervised local learning beyond classification with long-term feature bank. *arXiv preprint arXiv:2406.00446*, 2024.

A Appendix Roadmap

The appendix is organized as follows. Appendix B discusses related work on local learning and memory-efficient LLM post-training. Appendix C provides additional training and evaluation details. Appendix D provides the full SFT tables for all four SFT datasets and all three model families, including the Tulu-3 and Magpie-pro settings not shown in the representative main-text SFT table (Table 1). Appendix E provides the full GRPO results for GSM8K and NuminaMath, including the complete per-benchmark tables behind the matched-model GSM8K slice in Table 2. The main text reports the mechanism diagnostics, including freeze-k_1, $k = 2$ vs. $k = 4$, and front-layer perturbation in Tables 4, 5, and 3. Appendix F expands the efficiency analysis with matched-stack profiling, pipeline-parallel layout checks, and single-/dual-GPU scaling results. Appendix G reports LoRA quality results. Appendix H gives an additional 32B GRPO sanity check. Appendices I–M document the original study's implementation, metrics, compute, and auxiliary network. Appendices N–P report the additional configuration-level assessment, schedule diagnostics, and implementation checks.

B Related Work

B.1 Local learning.

Local learning has been studied as an alternative to standard end-to-end backpropagation by replacing a single global objective with local training signals. A central motivation is to reduce backward dependencies and alleviate locking phenomena, where earlier modules must wait for later modules to complete forward and backward computation before being updated. Decoupled neural interfaces and synthetic gradients address this issue by predicting gradient information from local activations (Jaderberg et al., 2017). Supervised local learning further partitions deep networks into blocks or layers trained with auxiliary objectives, reducing activation storage and enabling more localized updates (Belilovsky et al., 2020; Nøkland & Eidnes, 2019). Recent methods improve local learning with stronger auxiliary losses and better cross-block communication, narrowing the performance gap with end-to-end training (Su et al., 2024a;b; Zhang et al., 2025; Zhu et al., 2024). Information-preserving local objectives also predate LoPT. For example, InfoPro combines reconstruction with a cross-entropy or contrastive term to train gradient-isolated modules (Wang et al., 2021). LoPT builds on this local-learning perspective but uses a persistent, asymmetric objective assignment during LLM post-training: the first half receives reconstruction updates without local task supervision, while the second half optimizes the post-training task. The contribution is this instantiation and its measured adaptation–efficiency behavior, not the invention of reconstruction or gradient isolation.

B.2 Memory-efficient LLM post-training.

Memory-efficient post-training has been widely studied through algorithmic changes to how large language models are adapted. Parameter-efficient tuning methods reduce the number of trainable parameters by updating only small task-specific modules or low-rank adapters, including adapters, prefix tuning, prompt tuning, and LoRA (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Hu et al., 2022). Quantized fine-tuning further lowers memory usage by combining low-rank adaptation with low-bit pretrained weights, as in QLoRA (Dettmers et al., 2023). More recent optimizer-efficient methods reduce training memory from the optimization side, for example by projecting gradients into low-rank subspaces or reducing optimizer-state overhead (Zhao et al., 2024). Selective and blockwise optimization are also closely related. LISA samples layers for optimization while freezing most middle layers (Pan et al., 2024), and BAdam uses block-coordinate optimization with Adam updates (Luo et al., 2024). These approaches change which parameters are active in task optimization. LoPT instead retains a persistent first-half task-gradient boundary and assigns a separate reconstruction objective below it. This distinction does not establish superiority over these methods; direct matched comparisons are outside the reported study.

C Additional Experimental Details

C.1 Training Details

The original main SFT training runs use the eight-A100 setup. The single-GPU, dual-GPU, and pipeline-parallel profiles use the configurations specified in Appendix F. GRPO rollout and policy-update placement follow Appendix J. The additional assessment uses the separately specified configurations in Appendix N. For SFT, we use a per-device batch size of 4 and a learning rate of 2×10^{-5}. LoPT uses a default auxiliary reconstruction weight of $\lambda_{\text{aux}}=10.0$. For GRPO, each training group contains 4 prompts, with 4 sampled responses per prompt. We train for 1 epoch with learning rate 5×10^{-6}, temperature 0.7, top-$p = 0.95$, and clipping parameter $\varepsilon=0.2$.

C.2 Evaluation Details

For the original main-study results, we assess LoPT with lm-eval-harness (Gao et al., 2021) under a unified protocol. The main evaluation benchmark includes MMLU (5-shot) (Hendrycks et al., 2020), IFEval (Zhou et al., 2023), ARC-Challenge (25-shot) (Clark et al., 2018), GSM8K (4-shot) (Cobbe et al., 2021), HellaSwag (5-shot) (Zellers et al., 2019), TruthfulQA MC2 (Lin et al., 2022), and Winogrande (5-shot) (Sakaguchi et al., 2021). These benchmarks cover general knowledge, instruction following, reasoning, commonsense understanding, truthfulness, and mathematical problem solving. Appendix N lists the different prompt and metric protocol used by the additional assessment.

For training efficiency, we additionally report peak GPU memory during training and measure efficiency using throughput or update time, depending on the experiment.

D Full SFT Results

This section provides the full SFT results behind the main-text SFT discussion. Table 1 shows the representative SFT slice used in the main paper. Here we report the full SFT sweep across Alpaca-52K, Tulu-3 100K, Magpie-100k, and MetaMathQA 100K for all three model families. All rows use the same seven-benchmark evaluation suite as the main paper. The average deltas are unweighted averages over the seven reported benchmarks.

Base-model scores are reported once in Table 6; the per-dataset SFT tables below omit repeated base columns for compactness and focus on E2E–LoPT deltas.

Table 6: **Base-model scores used for SFT comparisons.** Values are percentages under the same seven-benchmark lm-eval protocol as the SFT tables.

Model	MMLU	IFEval	ARC-C	GSM8K	HellaSwag	TruthfulQA	Winogrande	Avg.
Qwen3-4B-Base	70.07	23.84	57.84	85.14	65.62	54.85	67.56	60.70
Qwen2.5-7B-Ins	74.17	57.30	66.72	83.47	79.35	64.75	74.51	71.47
Llama-3.1-8B-Ins	68.20	44.54	60.49	69.67	79.42	54.56	77.50	64.91

Table 7: **SFT training statistics** for all 12 SFT configurations. Memory is peak memory per GPU. Throughput is total training throughput in tokens/s. Parenthetical memory changes are calculated from the displayed memory values and rounded to one decimal place.

		Peak Mem (GB/GPU)		Throughput (tok/s)		Time (s)	
Dataset	Model	E2E	LoPT	E2E	LoPT	E2E	LoPT
Alpaca 52K	Qwen3-4B-Base	31.5	21.9 (−30.5%)	21,020	21,469	3,800	3,720
	Qwen2.5-7B-Ins	57.7	37.2 (−35.5%)	15,561	15,971	5,133	5,001
	Llama-3.1-8B-Ins	60.6	38.9 (−35.8%)	14,483	14,882	5,515	5,367
Tulu-3 100K	Qwen3-4B-Base	42.6	32.9 (−22.8%)	25,323	30,162	8,087	6,790
	Qwen2.5-7B-Ins	63.5	45.5 (−28.3%)	21,291	23,951	9,619	8,551
	Llama-3.1-8B-Ins	64.1	45.2 (−29.5%)	19,185	21,747	10,675	9,418
Magpie 100K	Qwen3-4B-Base	33.0	25.2 (−23.6%)	24,963	26,571	4,102	3,854
	Qwen2.5-7B-Ins	58.7	38.4 (−34.6%)	19,345	20,581	5,293	4,975
	Llama-3.1-8B-Ins	61.4	39.9 (−35.0%)	17,833	19,083	5,742	5,366
MetaMathQA	Qwen3-4B-Base	33.0	25.2 (−23.6%)	25,019	26,670	4,093	3,840
	Qwen2.5-7B-Ins	58.7	38.4 (−34.6%)	19,567	20,511	5,233	4,992
	Llama-3.1-8B-Ins	61.4	39.9 (−35.0%)	18,028	19,194	5,680	5,335

Table 8: **Alpaca-52K SFT results** with per-benchmark deltas (%).

	Qwen3-4B-Base			Qwen2.5-7B			Llama-3.1-8B		
Benchmark	E2E	LoPT	Δ	E2E	LoPT	Δ	E2E	LoPT	Δ
MMLU	**70.12**	69.99	−0.13	73.09	**74.08**	+0.99	**68.67**	68.48	−0.19
IFEval	25.87	**29.39**	**+3.52**	40.30	**41.22**	+0.92	30.50	**46.03**	**+15.53**
ARC-C	61.77	**61.95**	+0.18	60.58	**63.96**	**+3.38**	57.59	**60.41**	+2.82
GSM8K	79.45	**84.62**	**+5.17**	74.22	**79.38**	**+5.16**	46.62	**72.10**	**+25.48**
HellaSwag	69.80	**69.97**	+0.17	78.55	**79.46**	+0.91	78.31	**79.38**	+1.07
TruthfulQA	53.07	**53.91**	+0.84	52.28	**54.86**	+2.58	47.39	**52.67**	**+5.28**
Winogrande	**68.19**	67.69	−0.50	**75.77**	75.63	−0.14	76.09	**76.64**	+0.55
LoPT Wins	5/7, avg +1.32 pp			6/7, avg +1.97 pp			6/7, avg **+7.22** pp		

Table 9: **Tulu-3 100K SFT results** with per-benchmark deltas (%).

	Qwen3-4B-Base			Qwen2.5-7B			Llama-3.1-8B		
Benchmark	E2E	LoPT	Δ	E2E	LoPT	Δ	E2E	LoPT	Δ
MMLU	69.35	**69.80**	+0.45	73.64	**74.16**	+0.52	65.35	**67.54**	**+2.19**
IFEval	**21.63**	20.59	−1.04	33.09	**39.74**	**+6.65**	23.29	**37.34**	**+14.05**
ARC-C	60.75	**61.60**	+0.85	60.49	**63.40**	**+2.91**	57.94	**60.41**	**+2.47**
GSM8K	77.10	**79.45**	+2.35	51.55	**68.31**	**+16.76**	63.91	**74.83**	**+10.92**
HellaSwag	**70.93**	70.58	−0.35	79.08	**79.57**	+0.49	78.68	**79.19**	+0.51
TruthfulQA	53.14	**53.77**	+0.63	50.66	**55.51**	**+4.85**	48.83	**54.17**	**+5.34**
Winogrande	**69.53**	68.67	−0.86	**75.93**	74.51	−1.42	**78.22**	77.11	−1.11
LoPT Wins	4/7, avg +0.29 pp			6/7, avg **+4.39** pp			6/7, avg **+4.91** pp		

Table 10: **Magpie 100K SFT results** with per-benchmark deltas (%).

Benchmark	Qwen3-4B-Base E2E	Qwen3-4B-Base LoPT	Qwen3-4B-Base Δ	Qwen2.5-7B E2E	Qwen2.5-7B LoPT	Qwen2.5-7B Δ	Llama-3.1-8B E2E	Llama-3.1-8B LoPT	Llama-3.1-8B Δ
MMLU	69.12	**69.69**	+0.57	73.28	**73.86**	+0.58	66.67	**67.80**	+1.13
IFEval	25.88	**29.39**	**+3.51**	36.23	**46.03**	**+9.80**	**46.21**	44.62	−1.59
ARC-C	61.77	61.77	0.00	62.20	**64.68**	**+2.48**	60.84	**61.77**	+0.93
GSM8K	79.45	**83.24**	**+3.79**	68.20	**80.44**	**+12.24**	**72.48**	71.89	−0.59
HellaSwag	**70.80**	69.98	−0.82	79.00	**79.69**	+0.69	78.08	**79.14**	+1.06
TruthfulQA	53.03	**53.81**	+0.78	**59.89**	59.80	−0.09	54.65	**54.66**	+0.01
Winogrande	**68.19**	66.69	−1.50	**74.90**	74.19	−0.71	76.48	**76.95**	+0.47
LoPT Wins	4/7, avg +0.90 pp			5/7, avg **+3.57** pp			5/7, avg +0.20 pp		

Table 11: **MetaMathQA 100K SFT results** with per-benchmark deltas (%).

Benchmark	Qwen3-4B-Base E2E	Qwen3-4B-Base LoPT	Qwen3-4B-Base Δ	Qwen2.5-7B E2E	Qwen2.5-7B LoPT	Qwen2.5-7B Δ	Llama-3.1-8B E2E	Llama-3.1-8B LoPT	Llama-3.1-8B Δ
MMLU	**70.18**	70.17	−0.01	74.13	**74.23**	+0.10	65.29	**67.94**	**+2.65**
IFEval	20.89	**22.26**	+1.37	**51.76**	51.46	−0.30	34.94	**43.99**	**+9.05**
ARC-C	63.57	**63.64**	+0.07	63.91	**64.33**	+0.42	58.70	**60.92**	**+2.22**
GSM8K	81.20	**81.21**	+0.01	86.80	**87.21**	+0.41	76.79	**79.27**	**+2.48**
HellaSwag	72.43	**72.53**	+0.10	80.85	**80.97**	+0.12	77.70	**79.71**	**+2.01**
TruthfulQA	54.49	**54.55**	+0.06	62.63	**62.66**	+0.03	52.06	**52.93**	+0.87
Winogrande	68.69	**68.88**	+0.19	71.51	**74.03**	**+2.52**	76.56	**76.64**	+0.08
LoPT Wins	6/7, avg +0.26 pp			6/7, avg +0.47 pp			7/7, avg +2.77 pp		

E FULL GRPO RESULTS

This section provides the full GRPO results behind the main-text GRPO discussion in Section 3.2.1. Table 2 includes the matched-model GSM8K GRPO slice used in the main text, while this appendix reports the complete GSM8K and NuminaMath GRPO tables. The original SFT and GRPO results in Appendices D and E share the main-study evaluation protocol. The additional assessment in Appendix N uses a different, separately documented protocol. The training-statistics table reports the measured GRPO policy-update configurations.

Table 12: **GRPO training statistics** for the measured GSM8K GRPO configurations. Memory is GB/GPU and step time is s/step for the policy-update phase. Relative reduction is $(t_{\mathrm{E2E}} - t_{\mathrm{LoPT}})/t_{\mathrm{E2E}}$; it is not the relative increase in steps per second.

		Peak Mem (GB/GPU)		Step Time (s)		Update-time reduction	
Dataset	Model	E2E	LoPT	E2E	LoPT	Abs. (s)	Rel. (%)
GSM8K	Qwen3-4B-Base	42.55	32.89	1.42	1.32	0.10	7.0%
GSM8K	Qwen2.5-7B-Ins	75.39	59.66	1.71	1.56	0.15	8.8%
GSM8K	Llama-3.1-8B-Ins	62.62	47.54	2.10	1.65	0.45	21.4%

E.1 GSM8K GRPO

Table 13: **Qwen3-4B-Base GSM8K GRPO** — seven-benchmark results (%).

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	70.07	70.01	**70.10**	+0.09
IFEval	23.84	**24.03**	23.84	−0.19
ARC-C	57.84	62.03	**62.29**	+0.26
GSM8K	85.14	88.35	**88.63**	+0.28
HellaSwag	65.62	68.76	**69.95**	**+1.19**
TruthfulQA	54.85	**54.78**	54.62	−0.16
Winogrande	67.56	**67.17**	66.85	−0.32
LoPT Wins		4/7		Avg: +0.16 pp

Table 14: **Qwen2.5-7B-Ins GSM8K GRPO** — seven-benchmark results (%).

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	74.17	**74.22**	74.18	−0.04
IFEval	57.30	56.01	**59.70**	**+3.69**
ARC-C	66.72	**66.72**	66.38	−0.34
GSM8K	83.47	88.70	**89.09**	+0.39
HellaSwag	79.35	81.23	**81.40**	+0.17
TruthfulQA	64.75	64.52	**64.83**	+0.31
Winogrande	74.51	75.22	**75.40**	+0.18
LoPT Wins		5/7		Avg: +0.62 pp

Table 15: **Llama-3.1-8B-Ins GSM8K GRPO** — seven-benchmark results (%).

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	68.20	68.44	**68.55**	+0.11
IFEval	44.54	**47.69**	47.03	−0.66
ARC-C	60.49	**61.69**	61.43	−0.26
GSM8K	69.67	82.02	**82.40**	+0.38
HellaSwag	79.42	80.39	**80.47**	+0.08
TruthfulQA	54.56	**54.53**	54.11	−0.42
Winogrande	77.50	76.80	**77.35**	+0.55
LoPT Wins		4/7		Avg: −0.03 pp

E.2 NUMINAMATH GRPO

Table 16: **Qwen2.5-7B-Ins NuminaMath GRPO** — seven-benchmark results (%).

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	74.17	74.14	**74.21**	+0.07
IFEval	57.30	58.23	**60.26**	**+2.03**
ARC-C	66.72	**66.30**	65.87	−0.43
GSM8K	83.47	**86.58**	86.35	−0.23
HellaSwag	79.35	81.41	**81.63**	+0.22
TruthfulQA	64.75	64.16	**64.43**	+0.27
Winogrande	74.51	73.16	**74.98**	**+1.82**
LoPT Wins		5/7		Avg: +0.54 pp

Table 17: **Llama-3.1-8B-Ins NuminaMath GRPO** — seven-benchmark results (%).

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	68.20	67.25	**68.34**	**+1.09**
IFEval	44.54	42.88	**43.07**	+0.19
ARC-C	60.49	57.68	**61.01**	**+3.33**
GSM8K	69.67	81.97	**84.31**	**+2.34**
HellaSwag	79.42	79.71	**79.84**	+0.13
TruthfulQA	54.56	53.88	**53.96**	+0.08
Winogrande	77.50	77.03	**77.40**	+0.37
LoPT Wins		**7/7**		Avg: **+1.08** pp

F Extended Efficiency Analysis

This section expands the systems analysis in Section 3.2.1. The key point is that LoPT reduces task-backward activation lifetime, while gradient checkpointing reduces stored activations through recomputation. These mechanisms are complementary.

F.1 Matched-Stack Profiling

Table 18: **Compatibility with standard training stacks.** Memory in GB/GPU; speed Δ is relative throughput improvement over the matched E2E baseline.

Stack	Profiling setting	E2E Mem	LoPT Mem	Mem Saved	Speed Δ
Gradient Checkpointing	Qwen2.5-7B, 8×A100, bs=4, seq=1024	58.70 GB	**38.40 GB**	**−34.6%**	+5%
LoRA (rank=32)	Qwen2.5-7B, 8×A100, bs=4, seq=1024	68.37 GB	**43.85 GB**	**−35.9%**	+4%
DeepSpeed-ZeRO2	Qwen2.5-7B-Ins profiling, bs=4, seq=1024	58.90 GB	**49.91 GB**	−15.3%	+5%
DeepSpeed-ZeRO3	Qwen2.5-7B-Ins profiling, bs=4, seq=1024	62.44 GB	**44.32 GB**	**−29.0%**	+3%

Table 19: **Pipeline-parallel layout check.** Qwen3-4B, 2×A100, batch size 1, sequence length 2048; memory in GB/GPU and speed Δ relative to DDP E2E.

Method	Profiling setting	Peak Mem.	Mem Saved	Speed Δ
DDP E2E	Qwen3-4B, 2×A100, bs=1, seq=2048	49.6 GB	—	—
PP-LoPT	Qwen3-4B, 2×A100, bs=1, seq=2048	**23.2 GB**	**−53%**	**+22%**

F.2 Dual-GPU Efficiency

Table 20: **Dual-GPU efficiency** (2×A100 80GB, Qwen3-4B, bs=1, seq=2048). Mem. Eff. is defined as (tok/s/GPU)/(GB/GPU).

Method	Parallelism	Peak Mem (GB/GPU)	Throughput (tok/s/GPU)	Mem. Eff.
E2E	DDP	49.6	2,735	55.1
E2E+GC	DDP	31.5	2,256	71.6
LoPT	PP	**23.2**	**3,343**	**144.1**
LoPT+GC	PP	**13.6**	2,769	**203.6**

For the reported two-GPU configurations, PP-LoPT has approximately 22% higher throughput and 53% lower per-GPU peak memory than DDP E2E. LoPT+GC has approximately 3.7 times the reported throughput-per-memory ratio of DDP E2E. Because the parallel layouts differ, these are configuration-level comparisons rather than isolated estimates of the LoPT algorithm's contribution.

F.3 Single-GPU Scaling Across Batch and Sequence Length

Table 21: **Single-GPU scaling across configurations** (1×A100 80GB, Qwen3-4B). Format: peak memory / throughput.

Config	E2E	E2E+GC	LoPT	LoPT+GC
bs=1, seq=4096	61.8 GB / 2,840	25.6 GB / 2,307	42.2 GB / 3,717	25.2 GB / 3,061
bs=2, seq=2048	61.2 GB / 3,359	25.6 GB / 2,784	42.2 GB / 3,898	25.2 GB / 3,175
bs=2, seq=4096	OOM	35.1 GB / 2,578	63.5 GB / 4,360	32.9 GB / 3,532
bs=4, seq=2048	OOM	35.1 GB / 3,145	63.5 GB / 4,607	32.9 GB / 3,715
bs=8, seq=1024	OOM	35.1 GB / 3,528	63.5 GB / 4,753	32.9 GB / 3,825

In these measured single-GPU configurations, LoPT without GC has higher throughput than E2E with GC. Adding GC to LoPT further lowers peak memory and changes throughput. These measurements characterize the tested training stacks; they do not isolate total FLOPs, communication, or allocator effects.

F.4 Extended Dual-GPU Configurations

Table 22: **Extended dual-GPU configurations** (2×A100 80GB, Qwen3-4B). Format: peak memory / throughput.

Config	E2E (DDP)	E2E+GC (DDP)	LoPT (PP)	LoPT+GC (PP)
bs=1, seq=2048	49.6 GB / 2,735	31.5 GB / 2,256	23.2 GB / 3,343	**13.6** GB / 2,769
bs=2, seq=2048	68.7 GB / 3,350	33.0 GB / 2,761	33.8 GB / 4,294	**16.8** GB / 3,453
bs=2, seq=4096	OOM	42.6 GB / 2,568	55.1 GB / 4,607	**24.6** GB / 3,702
bs=4, seq=2048	OOM	42.6 GB / 3,144	37.9 GB / 4,563	**20.9** GB / 3,648
bs=8, seq=2048	OOM	61.8 GB / 3,334	37.9 GB / 4,710	**20.9** GB / 3,712

F.5 Post-hoc memory-threshold summary

We summarize the sequence-length-2048 rows of Table 22, retaining its batch-size convention.

Table 23: Largest reported batch size within the measured set $\{1, 2, 4, 8\}$ satisfying each per-GPU memory threshold at sequence length 2048. Each entry is batch size followed by reported peak memory in GB. A dash means that no measured configuration in that column satisfies the threshold; it does not prove general infeasibility. These are post-hoc filters of two-A100 80GB measurements, not measurements on 24GB/40GB devices or optima over all configurations.

Threshold	E2E (DDP)	E2E+GC (DDP)	LoPT (PP)	LoPT+GC (PP)
24 GB	–	–	1 (23.2)	8 (20.9)
40 GB	–	2 (33.0)	8 (37.9)	8 (20.9)
80 GB	2 (68.7)	8 (61.8)	8 (37.9)	8 (20.9)

This is a descriptive summary of a finite measured grid. Different parallel layouts need not have identical effective-batch semantics, and the table does not establish optimal throughput or quality under a common hardware budget.

G LoRA Quality Results

This section provides the quality results behind the compatibility discussion in Section 3.2.1. LoRA constrains update rank, while LoPT constrains task-gradient reach. The results below show that the two mechanisms can be combined.

Table 24: **LoPT+LoRA SFT** (Qwen2.5-7B, MetaMathQA 100K, LoRA rank=32, seven benchmarks, %).

Benchmark	Base	E2E+LoRA	LoPT+LoRA	Δ
MMLU	74.17	73.82	**74.16**	+0.34
IFEval	57.30	53.05	**55.45**	**+2.40**
ARC-C	66.72	64.68	**65.27**	+0.59
GSM8K	83.47	84.49	**85.13**	**+0.64**
HellaSwag	79.35	80.65	**81.15**	+0.50
TruthfulQA	64.75	61.58	**61.62**	+0.04
Winogrande	74.51	73.88	**74.03**	+0.15
LoPT Wins		**7/7**		Avg: +0.67 pp

Table 25: **LoPT+LoRA GRPO** (Qwen2.5-7B, GSM8K GRPO, LoRA rank=32, seven benchmarks, %).

Benchmark	Base	E2E+LoRA	LoPT+LoRA	Δ
MMLU	74.17	74.13	**74.23**	+0.10
IFEval	57.30	**59.15**	58.96	−0.19
ARC-C	66.72	66.04	**66.55**	+0.51
GSM8K	83.47	88.47	**88.92**	+0.45
HellaSwag	79.35	81.32	**81.34**	+0.02
TruthfulQA	64.75	**64.76**	64.75	−0.01
Winogrande	74.51	73.40	**73.56**	+0.16
LoPT Wins		5/7		Avg: +0.15 pp

The reported LoRA combinations have an average SFT difference of +0.67 pp and a near-parity GRPO difference of +0.15 pp. These results establish feasibility of the combination in the tested settings, but do not isolate low-rank regularization, clipping, or rollout effects as the cause of the observed differences.

H 32B GRPO Sanity Check

This section reports an additional 32B GRPO sanity check. It is not included in the 17 main configurations because the main quality sweep focuses on 4.0B–8.0B models. We therefore retain it only as a limited near-parity check, not as evidence about scaling behavior.

Table 26: **Qwen2.5-32B GSM8K GRPO** — seven-benchmark results (%). Averages are computed from unrounded lm-eval outputs.

Benchmark	Base	E2E-GRPO	LoPT-GRPO	Δ
MMLU	83.32	83.24	**83.34**	+0.10
IFEval	65.80	**66.17**	65.51	−0.66
ARC-C	74.06	74.23	**74.55**	+0.32
GSM8K	90.05	94.47	**95.04**	+0.57
HellaSwag	85.90	85.87	**86.24**	+0.37
TruthfulQA	65.54	65.73	**65.75**	+0.02
Winogrande	80.35	80.27	**80.71**	+0.44
LoPT Wins		6/7		Avg: +0.16 pp

At 32B, E2E and LoPT are near parity under GSM8K GRPO. LoPT is higher on 6 of 7 benchmarks, the average difference is +0.16 pp, and all benchmark-level differences are within one percentage point. We therefore interpret this result only as a sanity check that the midpoint boundary does not obviously degrade this setting, not as evidence about scaling.

I GRPO Implementation Details

Table 27: **GRPO hyperparameters and implementation details.**

Component	Detail
Optimizer	AdamW
Learning rate (k_1, k_2)	5×10^{-6}
Gradient clipping	max norm = 1.0
Training precision	bfloat16
Auxiliary loss weight λ_{aux}	10.0
Rollout engine	vLLM 0.7.3, tensor parallel
Rollout temperature	0.7
Rollout top-p	0.95
Max generation length	1,024 tokens
Training truncation	2,048 tokens
Generations per prompt (G)	4
Prompts per step (B)	4
Clip ε	0.2
KL penalty / reference model	None
Log-probability ratio	Sequence-level, sum over generated tokens
Advantage normalization	Mean/std within each prompt group
Per-sample backward	Yes, with immediate graph release
Reward (GSM8K)	Exact match of numerical answer
Reward (NuminaMath)	Exact match of numerical answer

GRPO uses chat-template formatted prompts, while the original benchmark evaluation uses lm-eval prompts. These training and evaluation protocols answer different questions and should not be numerically mixed.

J Exact Implementation Details

This appendix records the implemented LoPT step order and the GRPO-specific update details referenced in Section 3.

Boundary reuse and update scheduling. LoPT passes the detached first-half activation to k_2 without explicitly recomputing it after the local update. The auxiliary backward is completed before constructing the corresponding task graph, allowing the first-half graph to be released earlier. Microbatch backward operations are distinct from optimizer steps: with gradient accumulation, parameter updates and scheduler advances occur at update-window boundaries. Checkpoint-induced activation replay is separate from refreshing the boundary supplied to the task branch. Appendix O compares the profiled schedules. The auxiliary head is removed at inference.

Gradient routing and boundary reuse

1. Compute $\mathbf{h}_0 = \mathrm{Embed}(\mathbf{x})$ and $\mathbf{h}_1 = f_{k_1}(\mathbf{h}_0)$.
2. Form the cached task input $\widehat{\mathbf{h}}_1 = \mathrm{sg}(\mathbf{h}_1)$.
3. Compute $\lambda_{\text{aux}}\mathcal{L}_{\text{aux}}(\mathbf{h}_1, \mathrm{sg}(\mathbf{h}_0))$ and backpropagate it into θ_1 and ϕ.
4. After the corresponding local backward has completed, evaluate k_2 using the cached $\widehat{\mathbf{h}}_1$.
5. Compute the task loss and backpropagate it into θ_2 only.

This schematic specifies the gradient paths and the reused activation; it does not equate a microbatch with an optimizer step. In the tested accumulated implementation, each participating parameter group has one optimizer-state advance per effective batch, with one scheduler advance. Checkpoint replay may add layer evaluations during backward, but it is distinct from explicitly refreshing the boundary after the local update.

GRPO update details. For GRPO, rewards are converted to group-normalized advantages and optimized with a clipped policy objective during the policy-update phase. LoPT does not change reward computation, advantage normalization, clipping, or rollout; it changes only the policy-update backward pass by applying the midpoint gradient boundary there. Rollout still uses the current full model for generation.

Table 28 summarizes the reported implementation settings for the original study. Appendix N records the distinct settings used in the additional three-seed comparison.

Table 28: **Reported implementation settings for the original study.**

Component	Reported original-study setting
Model checkpoints	Qwen3-4B-Base, Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, and the Qwen2.5-32B sanity-check model.
Prompt formatting	GRPO rollout uses chat-formatted prompts; original benchmark evaluation uses the lm-eval protocol in Appendix K.
Evaluation harness	lm-eval-harness; task metrics and shot counts are listed in Table 29.
Precision and optimization	BF16 training; the original implementation description lists AdamW-family optimization with bitsandbytes 8-bit states; rollout engine vLLM 0.7.3.
Training repetitions	One run per original-study configuration unless stated otherwise; the additional assessment uses three training seeds.
LoPT partition	Two blocks with a midpoint task-gradient boundary.
Auxiliary objective	Feature reconstruction; weight 10.0; bottleneck width $d/4$; architecture in Appendix M.
Original SFT schedule	Batch size 4 per device; learning rate 2×10^{-5}; cosine schedule with 3% warmup; maximum gradient norm 1.0.
GRPO setup	Table 27; learning rate 5×10^{-6}, four responses for each of four prompts, clip 0.2, rollout temperature 0.7 and top-p 0.95.
LoRA	Rank 32 in the reported LoRA experiments.
Distributed and checkpointed profiles	Layout and memory measurements are listed in Appendix F; GRPO rollouts use tensor parallelism on 6–7 GPUs and policy updates on one GPU, as reported for the original implementation.
Datasets	Alpaca-52K, Tulu-3, Magpie-Pro, MetaMathQA, GSM8K, and NuminaMath; subsets are identified in their experiment tables.

The reported settings do not include all model and tokenizer revisions, original random seeds, software package versions, dataset snapshot and preprocessing identifiers, LoRA options beyond rank, or complete DeepSpeed and checkpointing configuration files. These details are not inferred from library defaults.

K Metric Conventions

The original main-study numerical results are extracted from lm-eval-harness JSON outputs. Table 29 lists the metric keys used for that seven-benchmark suite; Appendix N separately records the additional assessment protocol.

Table 29: **Benchmark-to-metric mapping.**

Benchmark	lm-eval Metric Key	Notes
MMLU	`acc,none`	5-shot
IFEval	`prompt_level_strict_acc,none`	0-shot, strict
ARC-Challenge	`acc_norm,none`	25-shot, length-normalized
GSM8K	`exact_match,strict-match`	4-shot, strict match
HellaSwag	`acc_norm,none`	5-shot, length-normalized
TruthfulQA MC2	`mc2,none`	0-shot
Winogrande	`acc,none`	5-shot

L Compute Budget

Table 30: **Compute budget for the original experiment set.** The approximately 1,144 A100 GPU-hours exclude the additional configuration-level assessment, whose complete GPU-hour total is not reported here.

Experiment Set	GPU-Hours (A100)
SFT profiling and efficiency analysis	$\sim$16
SFT training (12 configurations $\times$ 2 methods)	$\sim$480
SFT evaluation (24 checkpoints $\times$ 7 benchmarks)	$\sim$96
GRPO training (main GRPO, LoRA-GRPO, and diagnostics)	$\sim$480
GRPO evaluation	$\sim$48
Framework stacking profiling	$\sim$24
Original-study total	$\sim \mathbf{1{,}144}$

M Auxiliary Network Architecture Details

The reconstruction head is a bottleneck MLP:

$$\text{LayerNorm}(d) \to \text{Linear}(d, d/4) \to \text{GELU} \to \text{Linear}(d/4, d).$$

It is used only during training and discarded at inference.

For the displayed architecture with affine LayerNorm and biases in both linear layers, let $m = d/4$. The nominal auxiliary parameter count is $2dm + m + 3d$: $2dm$ linear weights, $m + d$ linear biases, and $2d$ LayerNorm parameters.

Table 31: **Nominal auxiliary-head parameter counts for the stated affine-LayerNorm, biased-linear architecture;** values in millions are rounded to two decimals.

Model	Hidden size d	Bottleneck m	Exact nominal count	Displayed
Qwen3-4B-Base	2560	640	3,285,120	3.29M
Qwen2.5-7B-Ins	3584	896	6,434,176	6.43M
Llama-3.1-8B-Ins	4096	1024	8,401,920	8.40M
Qwen2.5-32B	5120	1280	13,123,840	13.12M

For Qwen3-4B-Base, the nominal auxiliary-head count is 3.29M, about 0.08% of the total model parameters. The reported initialization for the linear layers is Xavier uniform with gain 0.1 and zero biases. A separate initialization setting for the LayerNorm affine parameters is not reported. These counts cover the auxiliary head only; any storage associated with separating tied input and output embeddings is a distinct parameterization choice and is not included in the auxiliary-head count.

N Additional Three-Seed Comparison

We compare E2E, freeze-k_1, and LoPT on Qwen2.5-7B-Instruct and MetaMathQA using three training seeds. This study uses the training and evaluation settings in Tables 32 and 33, which differ from those of the original experiments. Tables 34–36 report every method and seed.

N.1 Scope and protocol

E2E learning rate selection uses HPO seed 11, followed by confirmation runs with training seeds 1001–1003 for E2E, freeze-k_1, and LoPT. Checkpoints are selected by answer-token validation NLL at the registered evaluation cadence; the seven downstream tasks are not used for checkpoint selection. The answer-token restriction here defines the validation metric used for checkpoint selection; it is distinct from the training-loss position set specified for the original study in Eq. 10. The additional assessment uses Qwen2.5-7B-Instruct, MetaMathQA, and the model's chat template. Table 32 records the known differences from the original main-study protocol.

Table 32: **Protocol differences between the original experiments and the additional three-seed comparison.** The two sets of results are reported separately.

Item	Original main study	Additional assessment
TruthfulQA	MC2	MC1
GSM8K	4-shot strict match	CoT, 8-shot strict match
HellaSwag	5-shot	10-shot
MMLU	5-shot; original aggregation	57 subjects, 5-shot, weighted by subject size
IFEval / ARC-C / WinoGrande	0 / 25 / 5-shot	0 / 25 / 5-shot; registered metric keys
Prompt formatting	Original training and lm-eval configurations	Qwen chat template; separately specified evaluation protocol
Training repetitions	Single run per configuration unless stated otherwise	Three confirmation training seeds (1001–1003)
Checkpoint selection	Original experiment-specific rule	Lowest scheduled answer-token validation NLL
Hardware	Original main runs: A100 setup in Appendix C.1	H800; device count, interconnect, and parallel layout not reported
Memory definition	Original table-specific peak-memory measurements	Peak allocated/reserved GiB; sampled device peak reported separately for schedule profiles

The configurations in Table 33 differ in more than the lower-block objective.

Table 33: **Optimization settings for the additional three-seed comparison.** Learning rates, AdamW β_2, and warmup differ across methods.

Item	E2E	freeze-k_1	LoPT J2
Task-facing learning rate	1×10^{-5}	2×10^{-5}	1.5×10^{-5}
First-half learning rate / objective	1×10^{-5}, task	Frozen	1×10^{-6}, reconstruction
Auxiliary-head learning rate	N/A	N/A	2×10^{-5}
AdamW β_2	0.999	0.999	0.98
Warmup ratio	0.03	0.03	0
Gradient clipping	Global trainable parameters	Global trainable parameters	Threshold 1; exact grouping not established
Gradient checkpointing	Decoder layers 0–27	Decoder layers 14–27	C1 non-reentrant; exact grouping follows the run configuration
Training seeds	1001, 1002, 1003	1001, 1002, 1003	1001, 1002, 1003

The confirmation runs use sequence length 1024, microbatch size 4, gradient accumulation 8, a reported effective batch size of 32, and one training epoch. Validation and checkpoint saving occur at optimizer steps 781, 1562, 2343, and 3124, with an additional evaluation of the final export. The checkpoint with the lowest mean answer-token validation NLL is selected; ties select the earlier step. Benchmark evaluation uses BF16, batch size 1, evaluation seed 11, the Qwen chat template, and the complete registered task splits without a sample limit. MMLU aggregates its 57 subjects by subject size, while the seven task-level metrics are averaged with equal weight.

N.2 Complete benchmark and NLL outcomes

Table 34 presents the two different rankings produced by the assessment. E2E has the lowest selected-validation NLL, whereas freeze-k_1 has the highest seven-task mean. LoPT lies between them on both summaries. The standard deviations are sample standard deviations across the three training seeds.

Table 34: **Summary of the additional three-seed assessment.** Macro is the unweighted mean of the seven task scores, reported as a percentage; its standard deviation is in percentage points. NLL is measured on the validation data used for checkpoint selection. All uncertainties are sample standard deviations across the three training seeds.

Method	Selected-validation NLL ↓	Seven-task Macro (%) ↑
E2E	$\mathbf{0.137346 \pm 0.000038}$	60.7501 ± 0.1807
freeze-k_1	0.141267 ± 0.000218	$\mathbf{64.0075 \pm 0.2547}$
LoPT	0.139754 ± 0.000111	63.7021 ± 0.2710

Table 35: **All benchmark runs in the additional assessment.** Scores are fractions. The evaluation protocol differs from the original main study as listed in Table 32.

Method	Seed	MMLU	IFEval	ARC-C	GSM8K-CoT	HellaSwag	TQA-MC1	WinoGrande	Macro
E2E	1001	0.735294	0.709797	0.626280	0.316907	0.765883	0.450428	0.661405	0.609428
E2E	1002	0.734653	0.700555	0.617747	0.326763	0.766680	0.446756	0.657459	0.607230
E2E	1003	0.734653	0.704251	0.624573	0.309325	0.768273	0.445532	0.654301	0.605844
freeze-k_1	1001	0.731591	0.678373	0.602389	0.569371	0.750946	0.446756	0.683504	0.637562
freeze-k_1	1002	0.728956	0.713494	0.604096	0.597422	0.749651	0.433293	0.671665	0.642654
freeze-k_1	1003	0.732303	0.715342	0.600683	0.559515	0.750448	0.440636	0.681137	0.640009
LoPT	1001	0.732018	0.678373	0.616041	0.548901	0.758016	0.439412	0.679558	0.636046
LoPT	1002	0.730736	0.691312	0.612628	0.579227	0.753236	0.439412	0.674033	0.640084
LoPT	1003	0.731021	0.702403	0.611775	0.528431	0.755527	0.438188	0.677190	0.634934

Across paired seeds, LoPT exceeds E2E in Macro by 2.9520 pp on average; LoPT's selected-validation NLL is higher by 0.0024080 ± 0.0001178, where the uncertainty is the sample standard deviation of the three paired differences. Relative to freeze-k_1, LoPT has a lower NLL by 0.0015134 ± 0.0001085 but a lower Macro by 0.3054 pp. LoPT exceeds E2E in task means only on GSM8K-CoT and WinoGrande, and exceeds freeze-k_1 only on MMLU, ARC-C, and HellaSwag. A post-hoc mean over the six tasks excluding GSM8K-CoT is approximately 0.465 pp lower for LoPT than E2E; it is an arithmetic diagnostic, not a registered primary metric. Thus, the aggregate changes are not uniform task-level improvements.

Table 36: **Selected-validation answer-token NLL for all confirmation runs.** The validation set participates in checkpoint selection and is not described as an independent test set.

Method	Seed	Selected step	Selected-validation NLL
E2E	1001	2343	0.1373637303
E2E	1002	3124	0.1373022829
E2E	1003	3124	0.1373720760
freeze-k_1	1001	3124	0.1410447666
freeze-k_1	1002	3124	0.1412762020
freeze-k_1	1003	3124	0.1414813054
LoPT	1001	3124	0.1396359933
LoPT	1002	3124	0.1397701262
LoPT	1003	3124	0.1398560016

For the three-seed Qwen2.5-7B assessment, LoPT reduces the reported mean peak allocated memory by 11.0204% and peak reserved memory by 8.3593% relative to E2E. These percentages apply only to this additional configuration and measurement definition.

N.3 E2E LEARNING-RATE SEARCH

The search covers four learning rates on HPO seed 11. Training seeds 1001–1003 use the selected E2E rate. It does not include a search over training duration or regularization, and equal total search budgets across the final methods were not established.

Table 37: **MetaMathQA E2E learning-rate search on HPO seed 11.**

Learning rate	Best validation NLL	Best step	Selected
5×10^{-6}	0.1385595329	2343	No
1×10^{-5}	0.1370964381	3124	Yes
2×10^{-5}	0.1394214810	3124	No
4×10^{-5}	0.1520773188	3124	No

O EXECUTION SCHEDULE AND MEMORY DIAGNOSTICS

O.1 SCHEDULES AND MEASUREMENT SCOPE

The profile compares three schedules. SF-joint retains the local computation graph while constructing the task branch. SF-local-immediate completes the local backward before constructing the corresponding task graph and reuses the cached, detached boundary. Here, "immediate" describes the ordering of backward computation, not an optimizer step after every microbatch. Recompute-after-k_1-update instead refreshes the boundary with a new lower-block forward after the local update. Checkpoint-induced activation replay may occur in all applicable schedules and is distinct from this explicit boundary refresh.

The profile uses Qwen2.5-7B-Instruct, MetaMathQA, sequence length 1024, microbatch size 4, accumulation 8, BF16, non-reentrant checkpointing, and the common AdamW workload configuration. Allocated memory, sampled device memory, and input-token throughput are reported separately. The measurements are descriptive profiles: independent repeat counts, timing-window details, and allocator-reset scope are not fully reported, so no statistical uncertainty or single-component causal attribution is claimed. Layer-call entries count the instrumented events in the profiling window, not complete model forwards. This is not a GC-on/off factorial comparison; a complete quality comparison of staged training and post-update recomputation has not been established.

Table 38: **Execution-schedule profile under a common workload.** Allocated-memory and sampled-device peaks are distinct measurements; throughput is the logged input-token rate. Call counts refer to instrumented layer events.

Schedule	Steady allocated (GiB)	Sampled device (GiB)	Input tok/s	k_1/k_2 layer calls
SF-joint	66.120368	70.172119	2207.6271	16 / 16
SF-local-immediate	58.676545	65.404541	2227.0108	16 / 16
Recompute-after-k_1-update	58.476655	65.297119	1986.8032	24 / 16

Local-immediate execution reduces allocated memory by 11.26% relative to joint execution. Relative to recomputation, it has 12.09% higher measured input-token throughput and uses 0.200 GiB more allocated memory. The throughput ratio corresponds to 10.79% less time for equal counted input work, not a measured reduction in full-training wall-clock time. These are schedule comparisons under the profiled workload, not isolated E2E–LoPT effects.

O.2 FULL-TRAINING AND SECOND-ARCHITECTURE MEMORY

Table 39 reports memory measurements for the additional study. The Qwen2.5-7B values are means of peak memory across three training runs; the Llama-3.1-8B/Alpaca values come from a separate profile with no reported repeat count. These measurements use different configurations and memory

definitions from the original study. Freeze-k_1 memory is unavailable here, so the table cannot rank all three methods by quality and memory jointly. The roughly 11% allocated-memory reductions here and the original 24–36% reductions (Table 1) describe their respective configurations.

Table 39: **Additional memory measurements.** These values use different training configurations and measurement definitions from the original main-study tables.

Configuration	Method	Peak allocated (GiB)	Peak reserved (GiB)
Qwen2.5-7B / MetaMathQA	E2E	65.937984	68.996094
Qwen2.5-7B / MetaMathQA	LoPT	58.671327	63.228516
Llama-3.1-8B / Alpaca	E2E	63.881193	Not reported
Llama-3.1-8B / Alpaca	LoPT	56.300045	Not reported

P Implementation Checks and Evaluation Repeatability

P.1 Reference-based correctness checks

We compare the local-immediate and recomputation implementations with their respective numerical references. The 18 reported passing checks cover event order, gradient routing, optimizer and scheduler transitions, checkpoint replay, padding, zero-rate freezing controls, and export/reload behavior. Table 40 reports zero differences for the listed quantities in the tested cases. Gradient-norm agreement alone is not elementwise gradient equality, and each schedule is compared with its own reference.

Table 40: **Reference checks for the tested schedules.** Each column compares a schedule with its own reference.

Check	Local-immediate vs. own reference	Recompute vs. own reference
Auxiliary/task loss difference	0	0
k_1/k_2 gradient-norm difference	0	0
Maximum parameter difference	0	0
Maximum optimizer-state difference	0	0
Parameter optimizer state-step	1	1
Scheduler advance per effective batch	1	1
Gradients cleared	Yes	Yes

P.2 Fixed-checkpoint evaluator repeatability

Four evaluation repetitions use the same LoPT seed-1002 checkpoint. MMLU, ARC-C, GSM8K-CoT, HellaSwag, TruthfulQA-MC1, and WinoGrande are unchanged across these repetitions, while IFEval varies. The seven-task Macro has mean 0.6399516067, sample standard deviation 0.0001524558, and range 0.0002640613, or 0.02640613 percentage points. These are repeated evaluations of one checkpoint, not additional training seeds, and they do not define a noise bound for other methods or checkpoints.

P.3 Artifact integrity record

The artifact-integrity check covers 772 files totaling 157,287,398,587 bytes and reports no hash mismatches against the recorded manifests. It verifies artifact consistency, not the validity of the experimental conclusions.