Bi-LoRA: Efficient Sharpness-Aware Minimization for Fine-Tuning Large-Scale Models

*Yuhang Liu, *Tao Li, Zhehao Huang, Zuopeng Yang, Xiaolin Huang
*Equal contribution., School of Automation and Intelligent Sensing, Shanghai Jiao Tong University

Paper Code

Problem

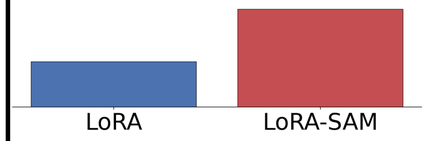
Given LoRA fine-tuning $W = W_0 + BA$,
our question is how to improve its generalization.

LoRA-SAM is Limited

$$\min_{B,A} \max_{|(\epsilon_B,\epsilon_A)|\le\rho} \mathcal{L}\big(W_0 + (B+\epsilon_B)(A+\epsilon_A)\big)$$

$$\begin{aligned}\epsilon_W &= B\epsilon_A + \epsilon_B A + \epsilon_B\epsilon_A \\ &\approx c[BB^\top(\nabla_W\mathcal{L}) + (\nabla_W\mathcal{L})A^\top A]\end{aligned}$$

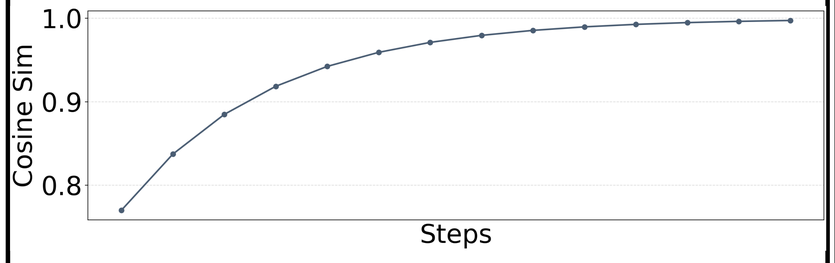
Issue 1. Doubling Computation



2 × Training Cost!

Another gradient step to solve inner maximization

Issue 2. Restricted Perturbation Space

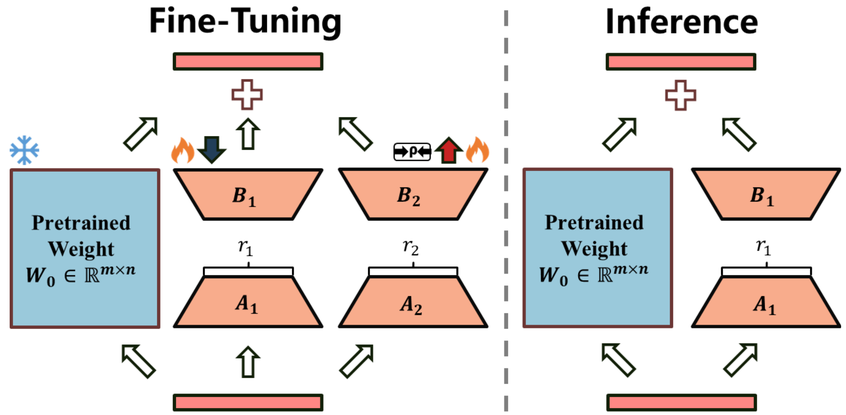


Early rise in cosine similarity indicates rapid subspace collapse.

- LoRA is merged into full weights at inference.
- LoRA-SAM perturbs only the LoRA subspace, i.e., ϵ_W is restricted in $Row(A)$ and $Col(B)$.
- Since this subspace converges quickly, sharpness optimization collapses early.

Can we achieve sharpness-aware optimization in a boarder space without sacrificing LoRA efficiency?

How Bi-LoRA Fixes LoRA-SAM



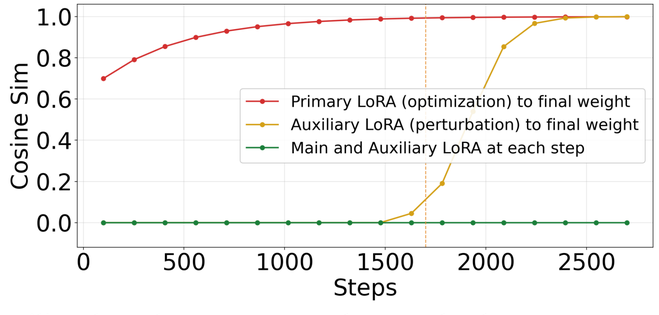
Fixing the Computation Bottleneck

$$\min_{B,A} \max_{|(\epsilon_B,\epsilon_A)|\le\rho} \mathcal{L}\big(W_0 + (B+\epsilon_B)(A+\epsilon_A)\big)$$ **Perturb then update**

$$\min_{B_1,A_1} \max_{|B_2A_2|_F\le\rho} \mathcal{L}(W_0 + B_1A_1 + B_2A_2)$$ **Parallel update**

- **One-pass update:** simultaneous **descent** on (A_1, B_1) and **ascent** on (A_2, B_2)
- **Standard inference:** discard the auxiliary branch at inference

Fixing the Perturbation Bottleneck



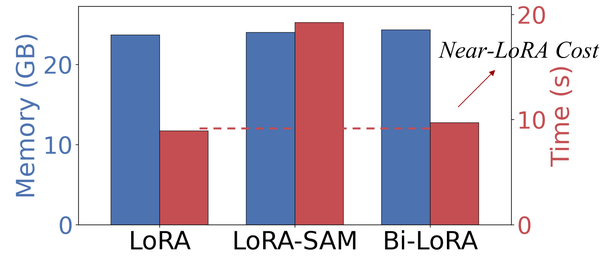
- The auxiliary branch converges much more slowly.
- A decoupled auxiliary branch carries the perturbation, instead of tying it to the primary optimization subspace
- **Broader perturbation exploration beyond the primary subspace.**

Experiments

- **Extensive experiments**: across Llama 2/3.1, Qwen 2.5-14B, and SDXL, covering math, code, chat, instruction following, NLU, and text-to-image generation.
- **Consistent generalization improvements**: +2.11 on GSM8K, +2.45 on HumanEval, +1.71 on DROP (Llama)
- **Ablations**: gains primarily come from adversarial ascent and decoupled perturbation.

Method	Llama 2-7B			Llama 3.1-8B			
	GSM8K	HumanEval	MT-Bench	MMLU	DROP	HEval	BBH
LoRA	58.21	24.75	5.92	63.38	49.82	43.15	42.82
LoRA-SAM	59.16	26.59	5.97	63.46	50.94	44.36	43.49
ROP	59.24	25.41	6.05	63.63	49.96	42.27	43.47
RWP_full	59.41	26.26	6.01	63.40	50.11	44.31	43.35
RWP_LoRA	58.81	24.92	5.80	63.50	50.16	42.68	**44.10**
LoRA-oBAR	59.26	26.30	5.97	63.62	49.92	43.49	43.44
LoRA-nBAR	59.72	26.50	6.10	63.45	49.80	45.23	43.39
Flat-LoRA	59.44	26.67	5.98	**63.67**	50.44	44.31	43.99
Bi-LoRA	**60.32**	**27.20**	**6.26**	**63.67**	**51.53**	**46.12**	43.45

LoRA-level Efficiency



- ~2 × faster than LoRA-SAM
- Only **minimal memory overhead.**

Takeaway

1. LoRA-SAM suffers from restricted and early-collapsing perturbations.
2. Bi-LoRA addresses this with a dual-LoRA design that decouples task adaptation and sharpness optimization
3. Near-LoRA cost. Standard LoRA inference. Better generalization than LoRA-SAM.