FZOO: Fast Zeroth-Order Optimizer for Fine-Tuning Large Language Models towards Adam-Scale Speed

Sizhe Dang[1*] Yangyang Guo[1*] Yanjun Zhao[1*] Xiaodong Zheng[1] Guang Dai[2] Ivor Tsang[3] Haishan Ye[1,2†]

[1]Xi'an Jiaotong University [2]SGIT AI Lab, State Grid Corporation of China [3]A*STAR

*Equal contribution †Corresponding author

Problem & Motivation

Fine-tuning LLMs with Adam consumes >10× inference memory (e.g., **633 GB** for OPT-30B). ZO methods like MeZO cut memory to inference level but converge ∼20× slower.

Can ZO methods achieve Adam-scale speed? Yes!

- **Adaptive step-sizes** via loss variance normalization
- **Rademacher (±1)** perturbations for efficient bit-level ops
- **Batched parallel** forward passes on GPU

FZOO vs. MeZO vs. Adam on RoBERTa-large

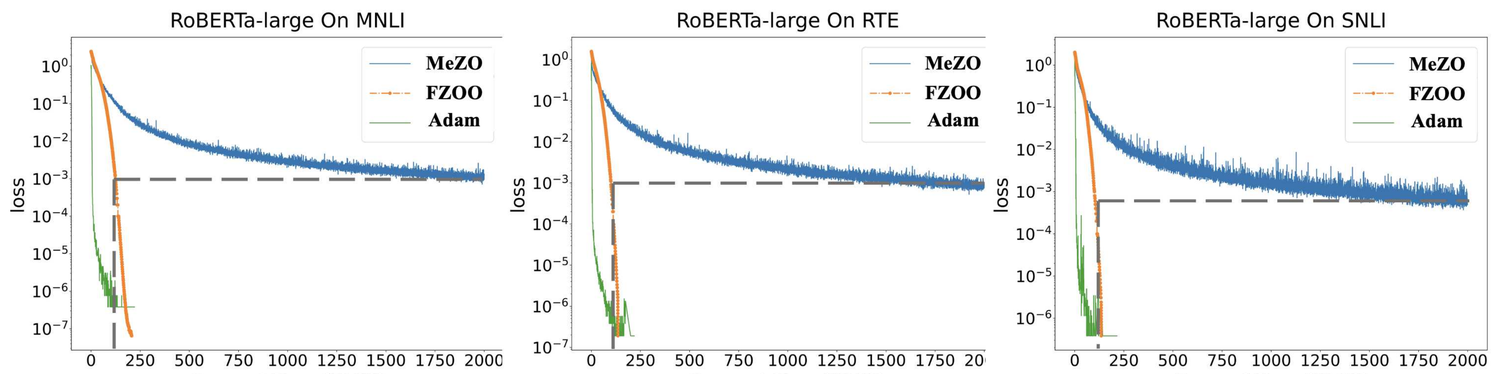

Figure 1. FZOO is **18×** faster than MeZO in forward steps, approaching Adam-scale convergence.

FZOO Method

Given N Rademacher perturbations $u_i \in \{+1, -1\}^d$:

$$l_i = L(\theta_t + \epsilon u_i; \mathcal{B}_t), \quad l_0 = L(\theta_t; \mathcal{B}_t)$$

Gradient estimate (one-sided, batched):

$$g_t = \frac{1}{\epsilon N} \sum_{i=1}^{N} (l_i - l_0)\, u_i$$

Adaptive step-size via loss standard deviation:

$$\sigma_t^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left(l_i - \bar{l}\right)^2$$

Update rule:

$$\boxed{\theta_{t+1} = \theta_t - \eta_t \frac{g_t}{\sigma_t}}$$

Large steps in flat regions (σ_t small), small steps in steep regions — Adam-style adaptivity at inference-level memory.

FZOO Pipeline

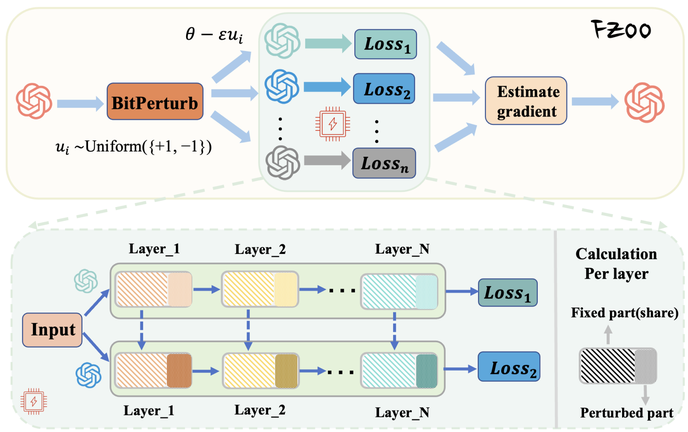

Figure 2. Batched one-sided estimates with Rademacher perturbations and parallel forward passes.

Theoretical Guarantee

Proposition. $\mathbb{E}[\sigma_t^2] = \epsilon^2 \|\nabla L\|^2 + O(\epsilon^3)$, so g_t/σ_t is a normalized stochastic gradient ⇒ FZOO ≡ normalized-SGD in ZO.

Theorem. Under $\mathcal{L}$-smoothness + bounded variance:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] = O\left(\frac{1}{\sqrt{T}}\right)$$

matching SGD's convergence rate for nonconvex optimization.

Memory Usage

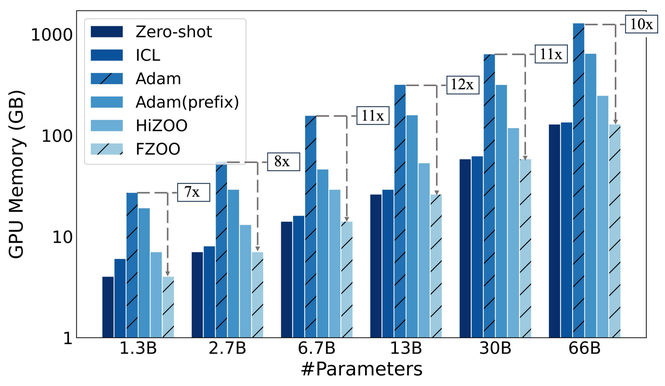

Figure 3. FZOO = inference-level memory. Adam+prefix still needs several × more.

Results: RoBERTa-large (350M, k=16)

Method	SST-2	SST-5	SNLI	MNLI	RTE	TREC	Avg
Zero-shot	79.0	35.5	50.2	48.8	51.4	32.0	49.5
FT (6×M)	91.9	47.5	77.5	70.0	66.4	85.0	74.9
HiZOO (2×M)	93.2	46.2	74.6	64.9	66.8	79.8	70.9
MeZO	90.5	45.5	68.5	58.7	64.0	76.9	67.4
FZOO	**93.3**	**47.6**	**75.9**	**64.9**	**67.9**	**78.8**	**71.4**

+5.9% avg over MeZO, **18×** fewer forward passes, comparable to HiZOO at **1×** memory.

Results: Auto-Regressive LLMs — Classification & NLI (1000 examples)

Model	Method	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC
Phi-2	Adam	84.4	61.6	79.0	68.8	59.6	67.7	77.8
Phi-2	MeZO	86.6	67.1	75.0	72.4	59.6	54.4	78.2
Phi-2	**FZOO**	**87.4**	**70.4**	**83.9**	**79.3**	**61.5**	**56.7**	**81.3**
Llama3	Adam	94.6	80.7	94.6	83.3	64.4	71.6	84.7
Llama3	MeZO	92.2	74.4	**69.6**	76.7	63.5	57.8	77.6
Llama3	**FZOO**	**94.3**	**77.6**	**69.6**	**81.8**	**65.4**	**60.8**	**81.5**
OPT-13B	Adam	92.1	79.1	71.4	77.0	63.5	69.6	76.2
OPT-13B	MeZO	91.4	66.1	66.0	67.6	63.5	59.4	57.3
OPT-13B	**FZOO**	**93.7**	**71.1**	**69.6**	**72.2**	63.5	**60.5**	**66.0**

Results: Auto-Regressive LLMs — MC & Generation

Model	Method	COPA	ReCoRD	SQuAD	DROP	Avg (11)
Phi-2	Adam	84.0	68.7	90.4	41.1	71.2
Phi-2	MeZO	**86.0**	71.7	85.7	**37.8**	70.7
Phi-2	**FZOO**	**86.0**	**72.0**	**86.7**	37.4	**73.0**
Llama3	Adam	89.0	86.9	89.7	58.4	81.6
Llama3	MeZO	88.0	**85.6**	86.7	**57.1**	75.4
Llama3	**FZOO**	**88.0**	85.3	**87.9**	56.5	**77.2**
OPT-13B	Adam	88.0	81.0	84.5	31.3	74.0
OPT-13B	MeZO	88.0	**81.7**	84.7	**30.9**	68.8
OPT-13B	**FZOO**	**87.0**	81.0	**84.8**	28.7	**70.7**

FZOO outperforms MeZO by avg **+2.75%** across all 11 tasks. Scales to **OPT-66B** (+2.43%).

Training Loss Curves across LLMs

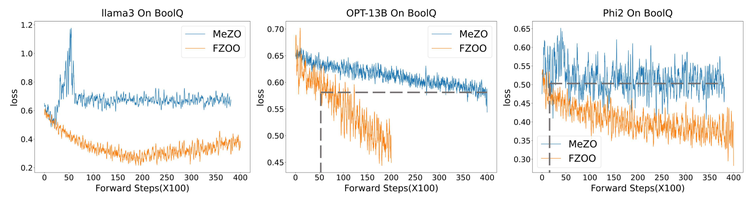

Figure 4. FZOO converges ∼8× faster than MeZO. Up to 7.4% accuracy gain on BoolQ.

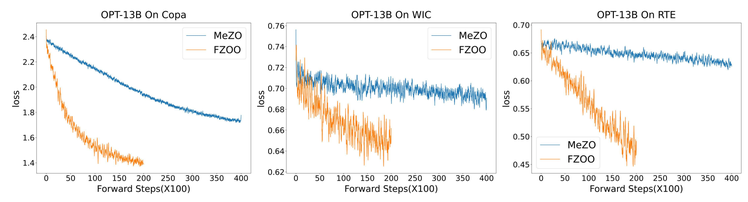

Figure 5. Loss curves on OPT family. FZOO consistently converges faster across model scales.

Speed Analysis

Task	SNLI	COPA	WIC	CB
FZOO speedup	20×	10×	9×	8×
w/ parallel	40×	20×	18×	16×

Adam commonly achieves 20× over MeZO. FZOO matches or exceeds this.

Comparison with ZO Variants

Method	Avg Acc	Memory	Runtime
ZO-SGD	88.4	1.0×	1.0×
ZO-Adam	86.1	2.47×	1.04×
HiZOO	90.5	2.04×	1.37×
FZOO	**90.4**	**1.0×**	**0.56×**

Matches HiZOO accuracy, **half the runtime**, **minimal memory**.

Efficient Batched Forward Pass

Rademacher (±1) perturbations turn matrix multiply into sign-flip additions — all N perturbations run in parallel via batch stacking:

$$Y^{(j)} = \underbrace{W^{(j)} Y^{(j-1)}}_{F^{(j)}} + \underbrace{\epsilon (U \odot Y^{(j-1)})}_{P^{(j)}}$$

Overall speedup: $\boxed{f \times \min(p, r)}$ On OPT-125M ($N = 8$): **1.92×** wall-clock gain.

Algorithm: FZOO

Require: $\theta \in \mathbb{R}^d$, loss L, steps T, scale ϵ, batch N, LR $\{\eta_t\}$

for $t = 1, \ldots, T$ **do**
  Sample batch $\mathcal{B} \subset \mathcal{D}$
  Compute $l_0 = L(\theta; \mathcal{B})$
  for $i = 1, \ldots, N$ **do**
    Sample seed s_i; generate $u_i \in \{+1, -1\}^d$
    $\theta \leftarrow \theta + \epsilon u_i$ {perturb}
    $l_i \leftarrow L(\theta; \mathcal{B})$
    $\theta \leftarrow \theta - \epsilon u_i$ {reset}
  end for
  $\sigma_t \leftarrow \mathrm{Std}(l_1, \ldots, l_N)$
  $\mathrm{grad}_i \leftarrow (l_i - l_0) \,/\, (N \cdot \sigma_t)$
  for $i = 1, \ldots, N$ **do**
    Regenerate u_i from s_i
    $\theta \leftarrow \theta - \eta_t \cdot \mathrm{grad}_i \cdot u_i$
  end for
end for

Takeaways

- **FZOO** = Adam-scale speed + inference-level memory for LLM fine-tuning
- Adaptive σ_t-normalization ≡ normalized-SGD in ZO regime, $O(1/\sqrt{T})$ convergence
- Plug-and-play with PEFT; scales 350M–**66B** on single GPU