北京大學
PEKING UNIVERSITY

Fine-tune LLMs with Zeroth-Order Optimization

Kun Yuan (袁 坤)

Center for Machine Learning Research @ Peking University

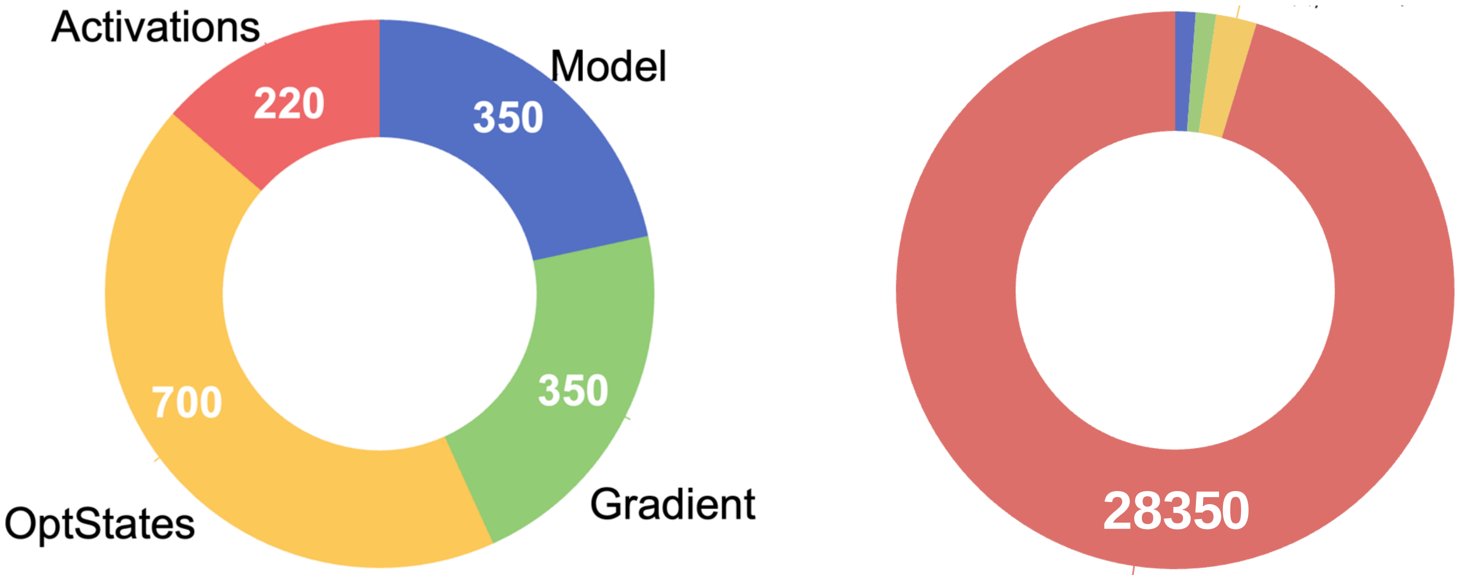
Activation dominates when sequence length is large

Memory = **Model** + **Gradient** + **Optimizer states** + **Activations**

Activation dominates when sequence length is large

GaLore/GoLore cannot save activations

GPT3-175B



Sequence length = 2048

Sequence length = 16348

Zeroth-order optimization

- Activations are auxiliary variables to facilitate the gradient calculations

Consider a linear neural network

$$z_i = X_i z_{i-1}, \forall\, i = 1, \cdots, L$$
$$f = \mathcal{L}(z_i; y)$$

The gradient is derived as follows

$$\frac{\partial f}{\partial X_i} = \frac{\partial f}{\partial z_i} z_{i-1}^{\top}$$

Need to store activations $z_1, z_2, \cdots, z_L$

- **If we do not calculate gradient, we do not need activations!**

- Can we fine-tune LLMs without calculating gradients? **Zeroth-order optimization**

Zeroth-order SGD

- Consider the **R**andom **G**radient **E**stimator (RGE)

$$\text{(RGE)} \quad \hat{\nabla} F(\boldsymbol{X};\xi) := \frac{F(\boldsymbol{X}+\epsilon \boldsymbol{Z};\xi) - F(\boldsymbol{X}-\epsilon \boldsymbol{Z};\xi)}{2\epsilon}\boldsymbol{Z}. \qquad \boldsymbol{Z} \sim \mathcal{N}(0,1)$$

- RGE approximates gradient with finite function value differences

- Zeroth-order stochastic gradient descent (ZO-SGD):

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^{t} - \alpha \hat{\nabla} F(\boldsymbol{X}^{t};\xi^{t}),$$

No need for backward propagation; Just forward passes; **No need to save gradients and activations!**

Memory-efficient Zeroth-Order (MeZO) algorithm

北京大学 PEKING UNIVERSITY

Algorithm 1: MeZO

Require: parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, loss $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$, step budget T, perturbation scale ϵ, batch size B, learning rate schedule $\{\eta_t\}$

for $t = 1, ..., T$ **do**
  Sample batch $\mathcal{B} \subset \mathcal{D}$ and random seed s
  $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$
  $\ell_+ \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$
  $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, -2\epsilon, s)$
  $\ell_- \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$
  $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$ ▷ `Reset parameters before descent`

  `projected_grad` $\leftarrow (\ell_+ - \ell_-)/(2\epsilon)$
  Reset random number generator with seed s ▷ `For sampling` z
  for $\theta_i \in \boldsymbol{\theta}$ **do**
    $z \sim \mathcal{N}(0, 1)$
    $\theta_i \leftarrow \theta_i - \eta_t *$ `projected_grad` $* z$
  end
end

Subroutine `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$
  Reset random number generator with seed s
  for $\theta_i \in \boldsymbol{\theta}$ **do**
    $z \sim \mathcal{N}(0, 1)$
    $\theta_i \leftarrow \theta_i + \epsilon z$
  end
  return $\boldsymbol{\theta}$

No need to store Z, $X + \epsilon Z$ and $X - \epsilon Z$. No need to compute gradient and optimizer states

MeZO: Memory-efficient Zeroth-Order Optimization

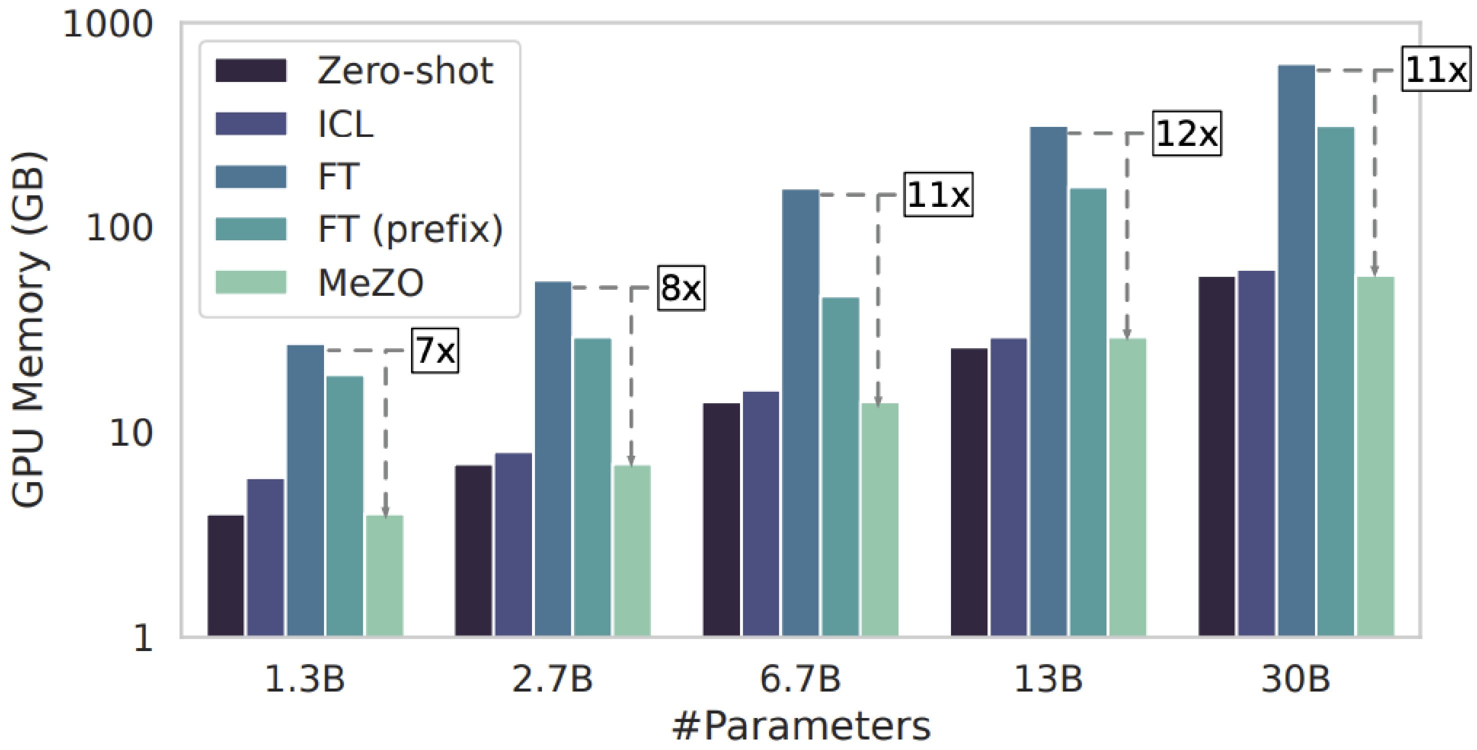


Figure 3: GPU memory consumption with different OPT models and tuning methods on MultiRC (400 tokens per example on average).

Sadhika Malladi et. al., *Fine-Tuning Language Models with Just Forward Passes*, NeurIPS 2023

MeZO: Memory-efficient Zeroth-Order Optimization

Method	Zero-shot / MeZO	ICL	Prefix FT	Full-parameter FT
1.3B	1xA100 (4GB)	1xA100 (6GB)	1xA100 (19GB)	1xA100 (27GB)
2.7B	1xA100 (7GB)	1xA100 (8GB)	1xA100 (29GB)	1xA100 (55GB)
6.7B	1xA100 (14GB)	1xA100 (16GB)	1xA100 (46GB)	2xA100 (156GB)
13B	1xA100 (26GB)	1xA100 (29GB)	2xA100 (158GB)	4xA100 (316GB)
30B	1xA100 (58GB)	1xA100 (62GB)	4xA100 (315GB)	8xA100 (633GB)
66B	2xA100 (128GB)	2xA100 (134GB)	8xA100	16xA100

Table 22: Memory usage on the MultiRC (avg #tokens=400) dataset.

Sadhika Malladi et. al., *Fine-Tuning Language Models with Just Forward Passes*, NeurIPS 2023

MeZO: Memory-efficient Zeroth-Order Optimization

Fine-tuning LLMs

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP
Task type	classification							multiple choice		generation	
Zero-shot	58.8	59.6	46.4	59.0	38.5	55.0	46.9	80.0	81.2	46.2	14.6
ICL	87.0	62.1	57.1	66.9	39.4	50.5	53.1	87.0	**82.5**	75.9	29.6
LP	**93.4**	68.6	67.9	59.3	63.5	60.2	63.5	55.0	27.1	3.7	11.1
MeZO	91.4	66.1	67.9	67.6	63.5	**61.1**	60.1	**88.0**	81.7	**84.7**	30.9
MeZO (LoRA)	89.6	67.9	66.1	**73.8**	**64.4**	59.7	61.5	84.0	81.2	83.8	**31.4**
MeZO (prefix)	90.7	**70.8**	**69.6**	73.1	60.6	59.9	**63.7**	87.0	81.4	84.2	28.9
FT (12x memory)	92.0	70.8	83.9	77.1	63.5	70.1	71.1	79.0	74.1	84.9	31.3

MeZO achieves comparable (within 1%) or better performance than FT on **7 out of 11 tasks**.

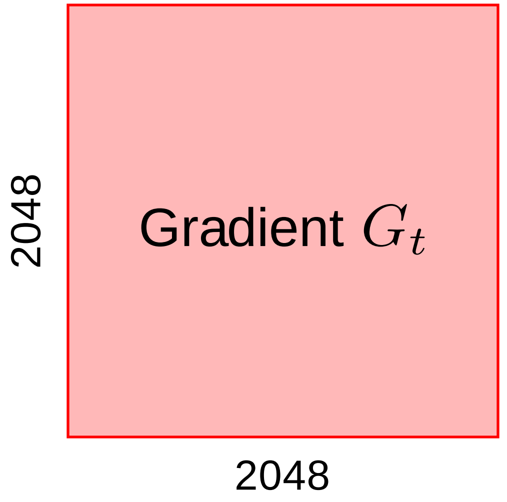
Sadhika Malladi et. al., *Fine-Tuning Language Models with Just Forward Passes*, NeurIPS 2023

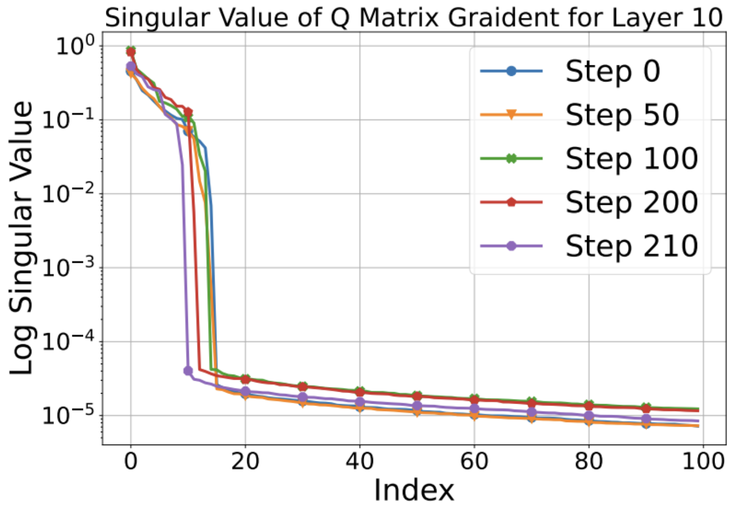
MeZO cannot approximate low-rank gradient

- Recall the **R**andom **G**radient **E**stimator (RGE)

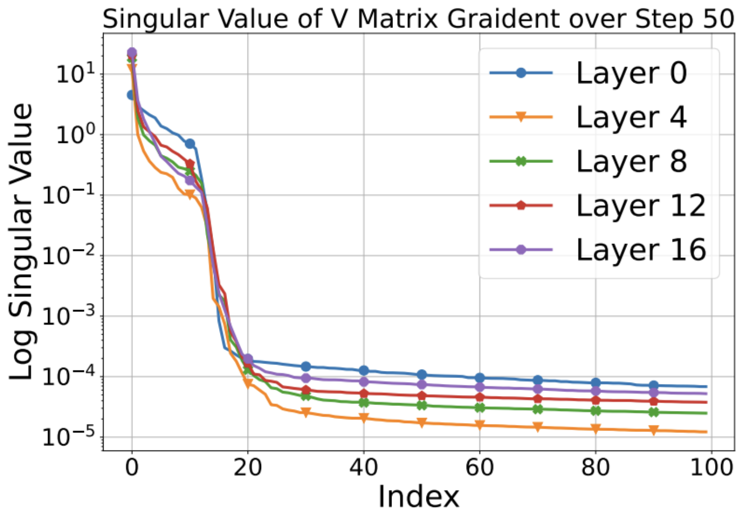
$$\text{(RGE)} \quad \hat{\nabla} F(\boldsymbol{X};\xi) := \frac{F(\boldsymbol{X}+\epsilon\boldsymbol{Z};\xi) - F(\boldsymbol{X}-\epsilon\boldsymbol{Z};\xi)}{2\epsilon}\boldsymbol{Z}. \qquad \boldsymbol{Z} \sim \mathcal{N}(0,1)$$

- Since $\boldsymbol{Z} \sim \mathcal{N}(0,1)$, it has almost **full-rank**. However, gradients in LLMs are low-rank





remain low-rank across iterations



remain low-rank across layers

Chen et. al., *Enhancing Zeroth-Order Fine-Tuning for Language Models with Low-Rank Structure*, 2024

Low-rank gradient estimator

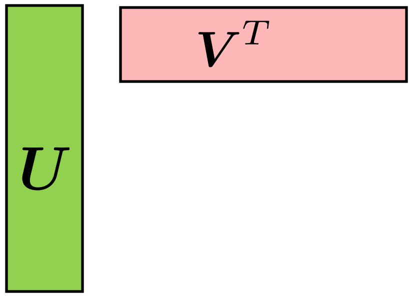
- RGE cannot approximate low-rank gradient; not proper for LLMs fine-tuning

$$\text{(RGE)} \quad \hat{\nabla} F(\boldsymbol{X};\xi) := \frac{F(\boldsymbol{X}+\epsilon \boldsymbol{Z};\xi) - F(\boldsymbol{X}-\epsilon \boldsymbol{Z};\xi)}{2\epsilon}\boldsymbol{Z}. \qquad \boldsymbol{Z} \sim \mathcal{N}(0,1)$$

- We propose a **L**ow-rank **G**radient **E**stimator (LGE)

$$\text{(LGE)} \quad \hat{\nabla} F(\boldsymbol{X};\xi) := \frac{F(\boldsymbol{X}+\epsilon \boldsymbol{U}\boldsymbol{V}^T;\xi) - F(\boldsymbol{X}-\epsilon \boldsymbol{U}\boldsymbol{V}^T;\xi)}{2\epsilon}(\boldsymbol{U}\boldsymbol{V}^T/\boldsymbol{r}).$$

where $\boldsymbol{U} \sim \mathcal{N}(0,1) \in \mathbb{R}^{m\times r}, \boldsymbol{V} \sim \mathcal{N}(0,1) \in \mathbb{R}^{n\times r}$



- Apparently, $\boldsymbol{U}\boldsymbol{V}^T$ is a low-rank matrix to approximate the true gradient

Chen et. al., *Enhancing Zeroth-Order Fine-Tuning for Language Models with Low-Rank Structure*, 2024

LOZO: Low-Rank Zeroth-Order SGD

- LGE estimator: $\mathrm{LGE}(\boldsymbol{X},\boldsymbol{U},\boldsymbol{V},\boldsymbol{r},\epsilon,\xi) := \dfrac{F(\boldsymbol{X}+\epsilon\boldsymbol{U}\boldsymbol{V}^T;\xi)-F(\boldsymbol{X}-\epsilon\boldsymbol{U}\boldsymbol{V}^T;\xi)}{2\epsilon}(\boldsymbol{U}\boldsymbol{V}^T/\boldsymbol{r}).$

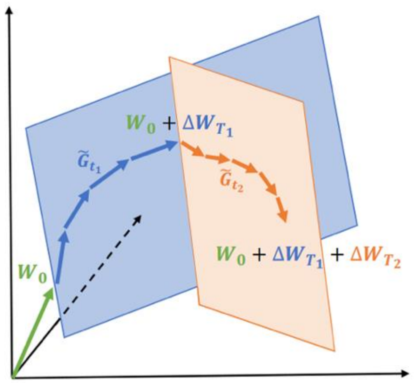
- LOZO vanilla version:

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \alpha\hat{\nabla}F(\boldsymbol{X}^t;\xi^t) \quad \text{where} \quad \hat{\nabla}F(\boldsymbol{X}^t;\xi^t) = \mathrm{LGE}(\boldsymbol{X}^t,\boldsymbol{U}^t,\boldsymbol{V}^t,\boldsymbol{r},\epsilon,\xi^t).$$

- If U^t and V^t are resampled per iteration, the subspace shifts too quickly; hurt performance

- Lazy sampling: explore each subspace sufficiently

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \alpha\hat{\nabla}F(\boldsymbol{X}^t;\xi^t), \quad \text{where} \quad \hat{\nabla}F(\boldsymbol{X}^t;\xi^t) = \mathrm{LGE}(\boldsymbol{X}^t,\boldsymbol{U}^t,\boldsymbol{V}^{(k)},\boldsymbol{r},\epsilon,\xi^t).$$



Sample V every τ iterations; search the space spanned by V

Chen et. al., *Enhancing Zeroth-Order Fine-Tuning for Language Models with Low-Rank Structure*, 2024

Convergence guarantees

Theorem 4.4. *Under Assumptions 4.1 and 4.2, and letting $T = K\nu$, with suitable choices of α and ϵ, the sequence of the $k\nu$-th variables $\{\boldsymbol{X}^{k\nu}\}$ generated by LOZO converges at the following rate:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\boldsymbol{X}^{k\nu})\|^2 \le O\left(\sqrt{\frac{\Delta_0 L\tilde{d}\sigma^2}{T}} + \frac{\Delta_0 L d\nu}{T}\right),$$

where $\Delta_0 := f(\boldsymbol{X}^0) - f^*$, $\tilde{d} = \sum_{\ell=1}^{\mathcal{L}}(m_\ell n_\ell^2/r_\ell)$ *and* $d = \sum_{\ell=1}^{\mathcal{L}} m_\ell n_\ell$.

LOZO learns the full-parameter even if we sample low-rank perturbations

LOZO converges at rate $O(1/\sqrt{T})$, the same rate as vanilla first-order SGD

Adding momentum incurs significant memory

- Momentum can significantly improve the convergence rate.

$$\boldsymbol{M}^t = \beta \boldsymbol{M}^{t-1} + (1-\beta)\hat{\nabla} F(\boldsymbol{X}^t; \xi^t), \quad \boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \alpha \boldsymbol{M}^t,$$

  But it incurs significant memory cost since M has the same dimension as ∇F

- However, we have $\nabla F = UV^T$ in LOZO. This structure can help us save memory

$$\boxed{\begin{aligned} \boldsymbol{M}^t &= \beta \boldsymbol{M}^{t-1} + (1-\beta)\boldsymbol{U}^t(\boldsymbol{V}^{(k)})^\top \\ \boldsymbol{X}^{t+1} &= \boldsymbol{X}^t - \alpha \boldsymbol{M}^t \end{aligned}} \Longrightarrow \boxed{\begin{aligned} \boldsymbol{N}^t &= \beta \boldsymbol{N}^{t-1} + (1-\beta)\boldsymbol{U}^t \\ \boldsymbol{X}^{t+1} &= \boldsymbol{X}^t - \alpha \boldsymbol{N}^t(\boldsymbol{V}^{(k)})^\top \end{aligned}}$$

Store M; Memory expensive

Store N; Memory efficient

Numerical performance

北京大学 PEKING UNIVERSITY

Memory consumption

Algorithm	MNLI		SNLI	
	Accuracy (%)	Memory Usage (GB)	Accuracy (%)	Memory Usage (GB)
LOZO	61.6	2.83	73.4	2.83
LOZO-M	**62.7**	2.84	**74.0**	2.84
MeZO	56.7	3.00	68.5	3.00
MeZO-M	58.9	5.89	69.6	5.89
MeZO-Adam	62.6	7.42	72.7	7.42

Numerical performance

Outperforms MeZO in most tasks

Task	SST-2	RTE	CB	BoolQ	WSC	WiC	MultiRC	COPA	ReCoRD	SQuAD	DROP
Zero-shot	58.8	59.6	46.4	59.0	38.5	55.0	46.9	80.0	81.0	46.2	14.6
ICL	87.0	62.1	57.1	66.9	39.4	50.5	53.1	87.0	**82.3**	75.9	29.5
MeZO	91.3	68.2	66.1	68.1	61.5	59.4	59.4	88.0	81.3	81.8	**31.3**
MeZO-LoRA	89.6	67.9	67.8	**73.5**	**63.5**	60.2	61.3	84.1	81.5	82.1	**31.3**
LOZO	**91.7**	**70.4**	**69.6**	71.9	**63.5**	**60.8**	**63**	**89.0**	81.3	**84.9**	30.7
FT	91.8	70.9	84.1	76.9	63.5	70.1	71.1	79.0	74.1	84.9	31.3

Table 2: Experiments on OPT-13B (with 1000 examples). ICL: in-context learning; FT: full fine-tuning with Adam. The best results are shown in **bold** except for FT.

Numerical performance

Tests with larger models

Task	SST-2	RTE	BoolQ	WSC	WiC	SQuAD
30B zero-shot	56.7	52.0	39.1	38.5	50.2	46.5
30B ICL	81.9	**66.8**	66.2	56.7	51.3	78.0
30B MeZO	90.7	64.3	68.2	63.5	56.3	**86.1**
30B LOZO	**92.8**	65.3	**72.3**	**64.4**	**57.2**	85.6
66B zero-shot	57.5	67.2	66.8	43.3	50.6	48.1
66B ICL	89.3	65.3	62.8	52.9	54.9	81.3
66B MeZO	92.0	71.5	73.8	**64.4**	57.8	84.0
66B LOZO	**92.5**	**74.0**	**74.5**	63.5	**59.4**	**85.8**

Table 3: Experiments on OPT-30B and OPT-66B on SuperGLUE benchmark. Our results show that LOZO is superior on most tasks compared to the other baselines. The best results are shown in **bold**.

Numerical performance

Training losses

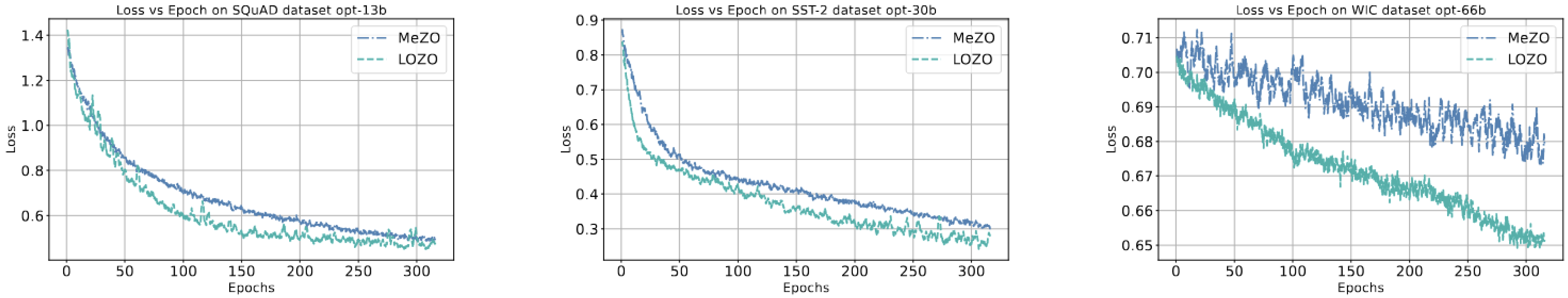


Figure 3: *Left:* Loss curves of OPT-13B on SQuAD dataset. *Middle:* Loss curves of OPT-30B on SST-2 dataset. *Right:* Loss curves of OPT-66B on WIC dataset.

Summary

- Activation dominates memory when batch size or sequence length is large
- Zeroth-order optimization can save activation
- MeZO cannot capture the low-rank structure in LLM gradients; we propose LOZO as an effective recipe
- LOZO outperforms MeZO and has strong theoretical guarantees

Take-home message

Low-rank structures are critical in LLMs pretraining and finetuning

北京大學
PEKING UNIVERSITY

Thank you!

Kun Yuan homepage: https://kunyuan827.github.io/