Efficient Training and Inference for Large Language Models

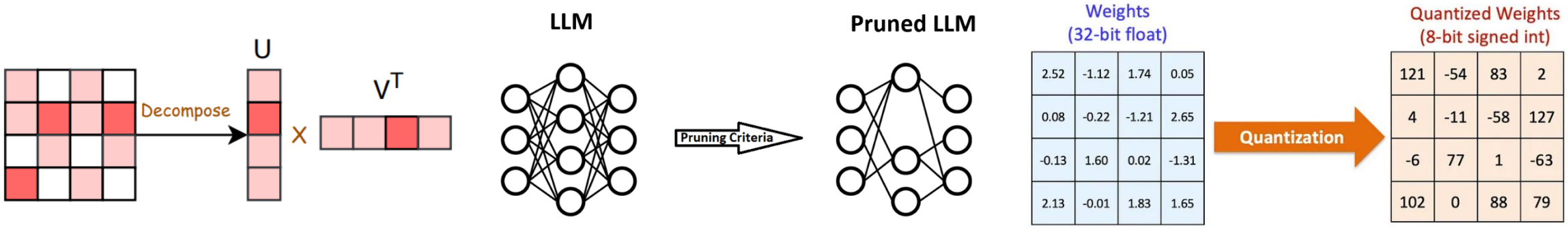
Yifan Yang (PhD Proposal)
Department of Computer Science
University of California, Santa Barbara

Table of Contents

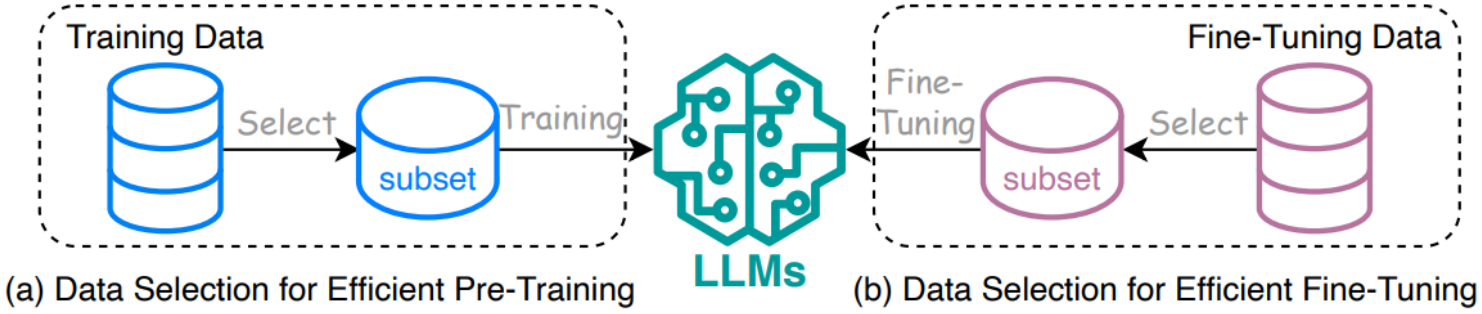
Table of Contents

There are many ways to leverage LLMs efficiently

Model-Centric Methods



Data-Centric Methods



(a) Data Selection for Efficient Pre-Training

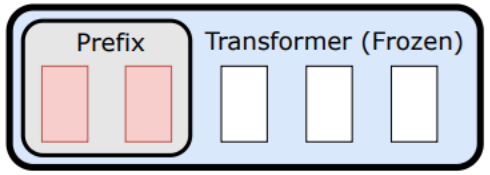
(b) Data Selection for Efficient Fine-Tuning

Efficient LLMs Fine-tuning

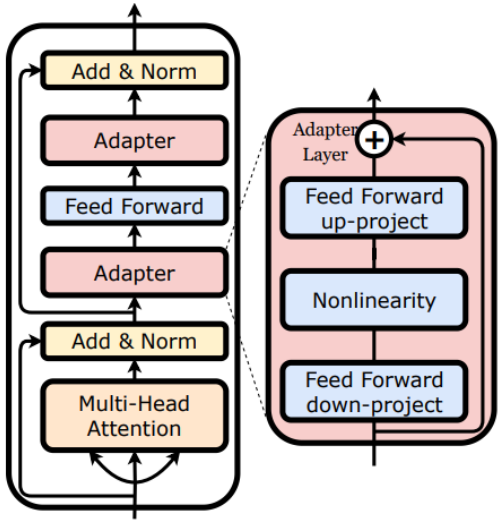
Parameter-Efficient Fine-tuning (PEFT)

- **Goal:** Perform fine-tuning on fewer parameters, while achieving performance on a downstream task that is comparable to fine-tuning of all parameters
- **Various approaches:**

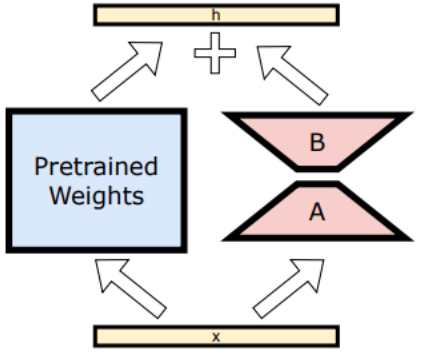
Prefix/Prompt Tuning



Series Adapters



LoRA



Efficient LLMs Fine-tuning

Zeroth-order Fine-tuning

- **Goal:** Fine-tuning the LLM with just forward pass, while achieving performance on downstream tasks that is comparable to gradient-based fine-tuning.
- Given the model weights θ, loss function $\mathcal{L}$ and minibatch data B , the ZO gradient estimation can be defined as:

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta};\mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{\theta}+\epsilon\boldsymbol{z};\mathcal{B}) - \mathcal{L}(\boldsymbol{\theta}-\epsilon\boldsymbol{z};\mathcal{B})}{2\epsilon}\boldsymbol{z} \approx \boldsymbol{z}\boldsymbol{z}^{\top}\nabla\mathcal{L}(\boldsymbol{\theta};\mathcal{B})$$

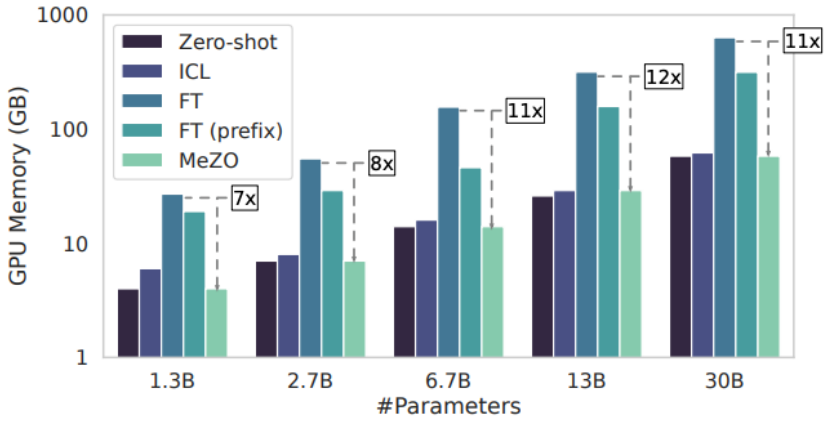
where z is random perturbation with same shape of θ and ϵ is a constant perturbation scale

Efficient LLMs Fine-tuning

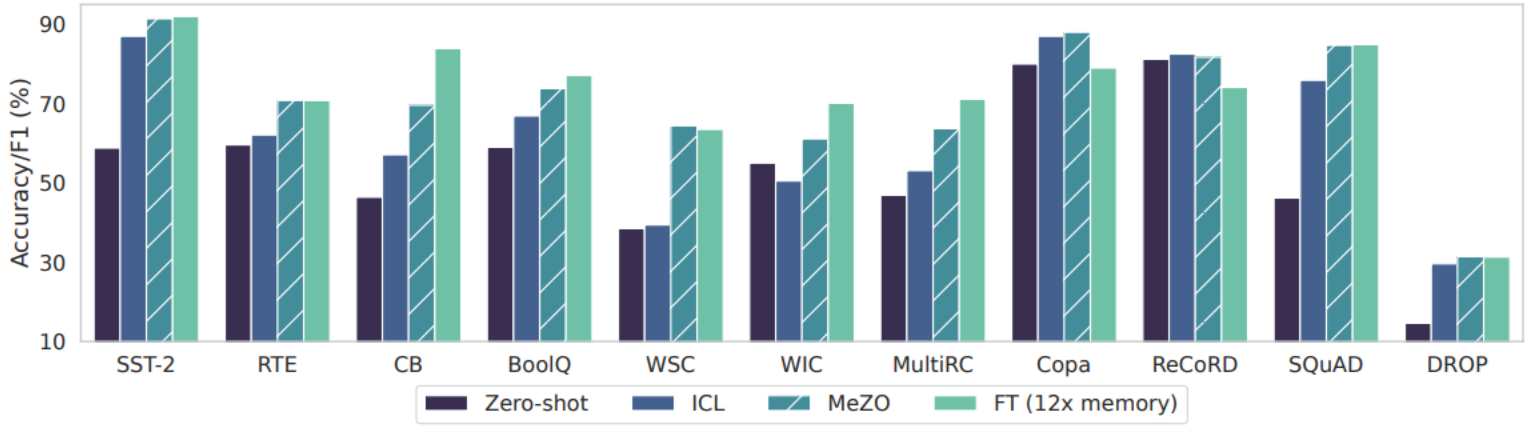
Zeroth-order Fine-tuning

- Zeroth-order Fine-tuning demonstrates strong performance reducing training memory by up to 12× compared to Adam-based fine-tuning [1].

Memory Comparison



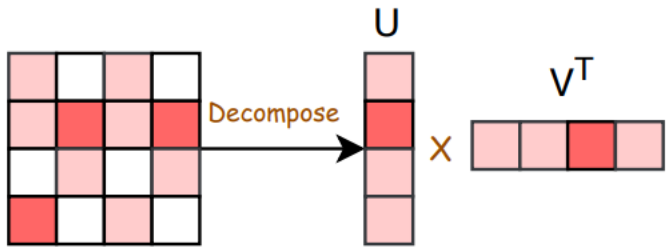
Performance Comparison



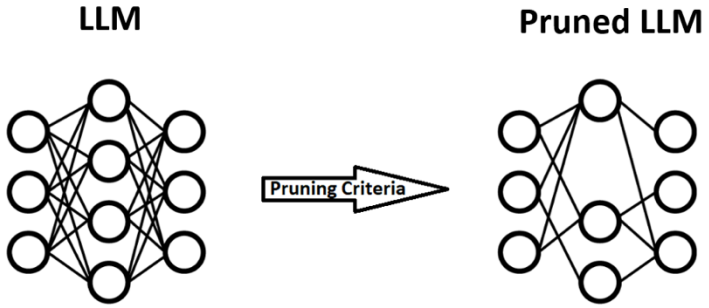
[1] Malladi, Sadhika, et al. "Fine-tuning language models with just forward passes." NeurIPS 2023.

Efficient LLMs Inference

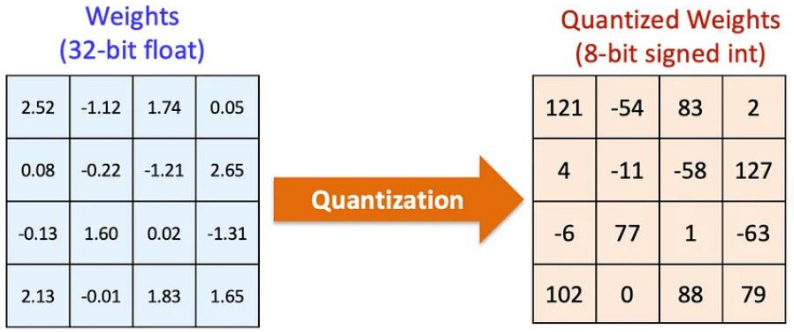
- Our focus is on model compression to accelerate LLM inference



Low-Rank Decomposition



Pruning



Quantization

	Low-Rank Decomposition	Pruning	Quantization
Goal	Factorize weights to lower-rank forms for efficiency.	Remove unimportant weights to reduce model size.	Lower precision of weights/activations.
Target	Weights (factorize)	Weights (sparsity)	Precision (bit-width)
Speedup	High (structured)	Moderate (Semi-structured) to high (structured)	High (hardware-ready)

Table of Contents

My Progress

- **Efficient Fine-tuning with Zeroth-order Optimization**
  - **Reducing Variance in Zeroth-Order Gradient Estimation via Parameter Reduction**

    Yifan Yang, et al, "LoRETTA: Low-Rank Economic Tensor-Train Adaptation for Ultra-Low-Parameter Fine-Tuning of Large Language Models", NAACL 2024.

    Yifan Yang , et al, "AdaZeta: Adaptive Zeroth-Order Tensor-Train Adaption for Memory-Efficient Large Language Models Fine-Tuning", EMNLP 2024.

  - **Sharpness-Aware ZO-VLM Prompt Tuning**

    Yifan Yang, et al. SharpZO: Hybrid Sharpness-Aware Vision Language Model Prompt Tuning via Forward-Only Passes, under review.

- **Efficient Inference with LLMs Pruning**
  - **Pruning the LLMs with regional gradients**

    Yifan Yang et al., "Wanda++: Pruning Large Language Models via Regional Gradients", in ACL Findings 2025.

Reducing Variance in Zeroth-Order Gradient Estimation via Parameter Reduction

Yifan Yang, et al, "LoRETTA: Low-Rank Economic Tensor-Train Adaptation for Ultra-Low-Parameter Fine-Tuning of Large Language Models", NAACL 2024.
Yifan Yang , et al, "AdaZeta: Adaptive Zeroth-Order Tensor-Train Adaption for Memory-Efficient Large Language Models Fine-Tuning", EMNLP 2024.

amazon

UC SANTA BARBARA

香 港 大 學
THE UNIVERSITY OF HONG KONG

Reducing the Variance of ZO Fine-tuning

Let's start from the convergence theory behind the ZO gradient estimation method given in:

$$\widehat{\nabla \mathcal{L}}(\theta_k; \mathcal{B}) = \sum_{q=1}^{Q_k} \frac{\mathcal{L}(\theta_k + \epsilon z_k^q; \mathcal{B}) - \mathcal{L}(\theta_k - \epsilon z_k^q; \mathcal{B})}{2\epsilon} z_k^q$$

Here, with approximation, we have a convergence rate of:

$$\mathbb{E}[\|\nabla \mathcal{L}(w_T)\|^2] \leq \mathcal{O}\left(\frac{R + \epsilon^2 L + C(d, \epsilon) \sum_k \frac{1}{q_k}}{K\epsilon}\right)$$

To reduce d, we propose a new **PEFT method LoRETTA**

To improve q_k, we propose an **adaptive query schedule**

where L is a constant related to the smoothness factor, $C(d, \epsilon)$is a term related to the model dimension d and q_k is the query at every step $k \in [1, K]$

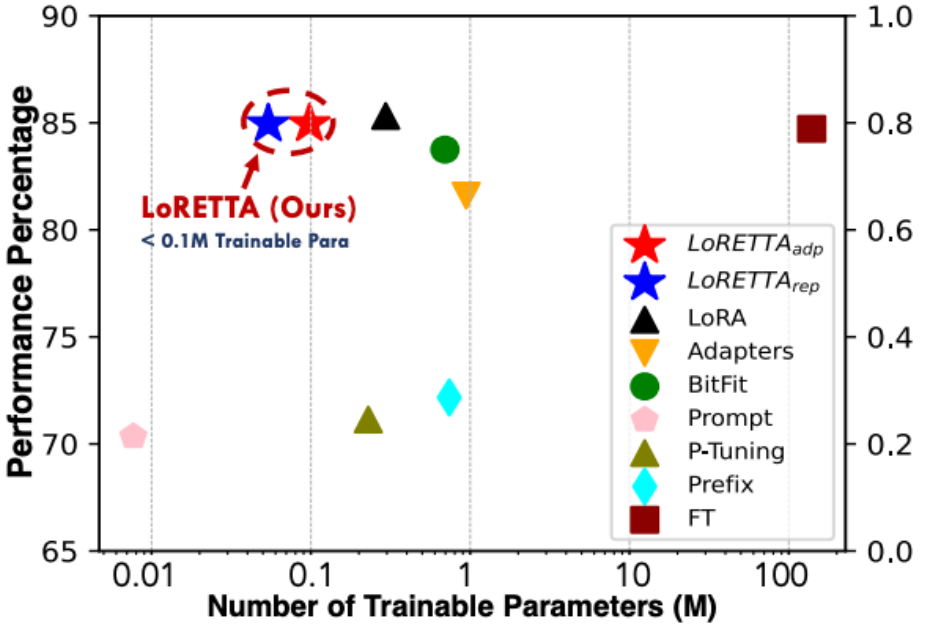
Fine-tuning LLMs with Even Fewer Parameters?

Compare with LoRA/Adapters, there exist ultra-low parameters solutions for parameter efficient fine-tuning of LLMs:

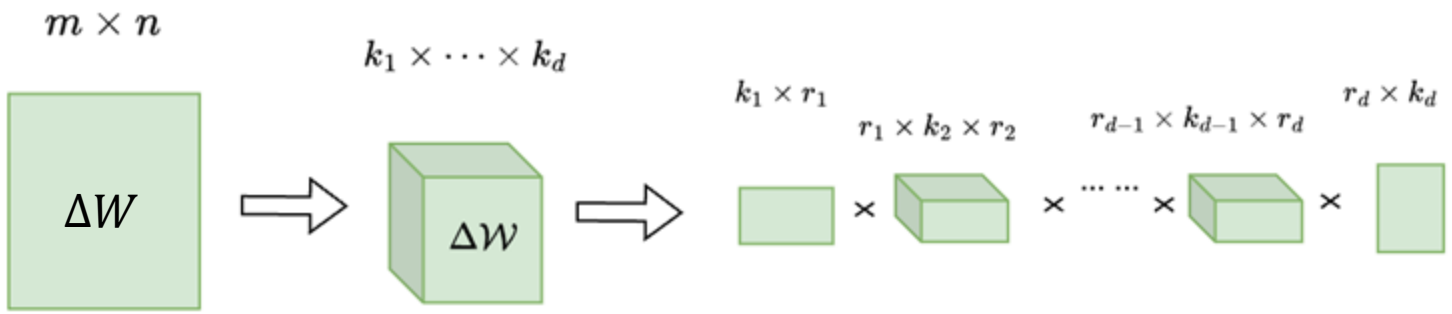
- **Rank reduction**: performance drop, still large number of param.
- **Bitfit** (Only fine-tuning bias): huge performance drop, fails to support recent models
- **Prompt-tuning**: huge performance drop

Can we significantly reduce the number of trainable parameters but keep high performance?

Introducing Low-Rank Economic Tensor-Train Adaptation (LoRETTA)



Tensorized Linear Layers



Decompose the weight matrix into list of tensor factors

- Step 1: Reshape the weight matrix into a d-way tensor
- Step 2: Decompose the d-way tensor into d 3-way tensor factors

Only tensor factors are stored and updated during the training process

The list of tensor factors will be contracted together during forward-pass

$$\mathcal{W} = \mathcal{G}_1 \times_{3,1} \mathcal{G}_2 \times_{3,1} \cdots \times_{3,1} \mathcal{G}_J,$$

LoRETTA Methods

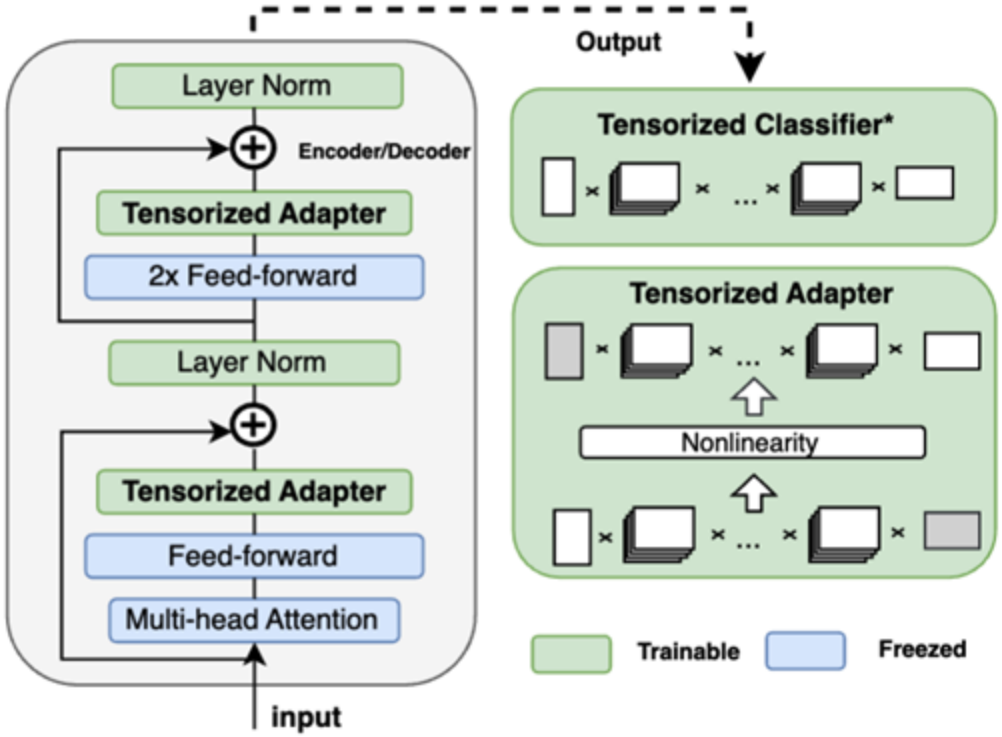
We apply the tensorized layer in the Adapters method and successfully reduce the number of trainable parameter up to 60X

LoRETTA

Add tensorized adapters after the FFN an Attn sub-layer

Tensorized the classifier(optional)

Trainable tensorized adapters, layer norm and last layer



Adaptive Query Schedule

Recall the ZO estimation equation:

$$\widehat{\nabla \mathcal{L}}(\theta_k; \mathcal{B}) = \sum_{q=1}^{Q_k} \frac{\mathcal{L}(\theta_k + \epsilon z_k^q; \mathcal{B}) - \mathcal{L}(\theta_k - \epsilon z_k^q; \mathcal{B})}{2\epsilon} z_k^q$$

- Instead of using a $Q_k = 1$ in previous LLMs fine-tuning papers, we find a sub-linear increasing query number balances the effectiveness and efficiency:

$$Q_k := \min(\alpha e_k^{\beta}, Q_{max})$$

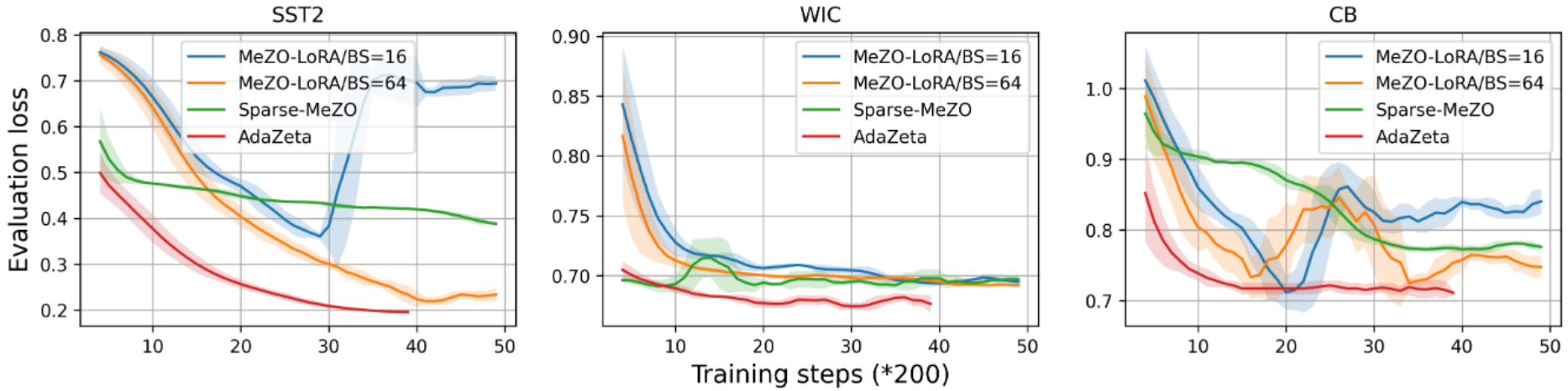
Where α, β are constants decided based on experiments, Q_{max} is a limited maximum query number and e is the epoch number

Experimental Results

We perform experiments on large scale 7B models

Experiment setup:

- Model: Llama-2-7B
- Data: Low data resource (1000 samples), prompting-based fine-tuning



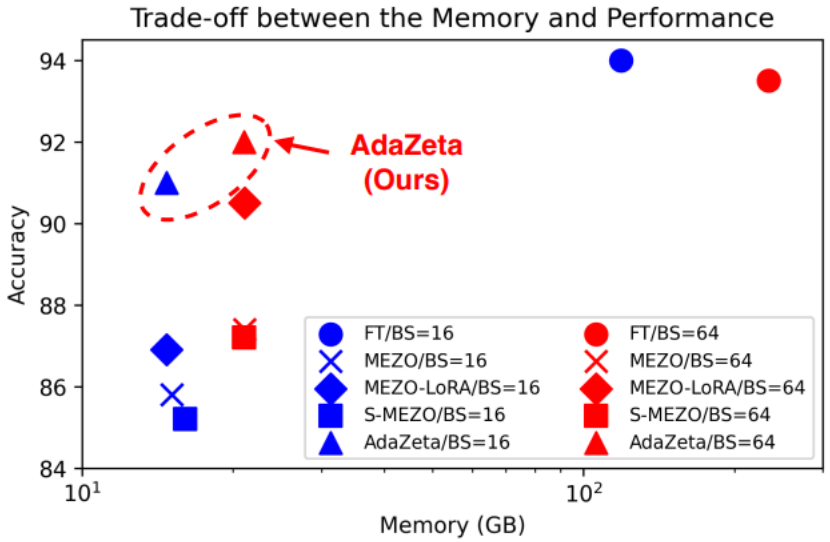
- The AdaZeta method successfully:
  - Speed up the convergence
  - Solve the divergence problem

Experimental Results

We also compare the time-to-test accuracy and the training memory to quantify the efficiency of our method:

Methods	SST2	WIC	CB	MultiRC
MeZO-LoRA(BS=64)	3.0	4.8	8,6	30.0
MeZO-LoRA(BS=16)	0.6	1.1	3.1	10.8
Sparse-MeZO	4.1	3.6	4.3	6.4
$\text{AdaZeta}_{\text{seq}}$	1.1	1.0	0.9	12.1
AdaZeta	0.9	0.9	0.8	10.6

Required GPU hours (GPU numbers × Training hours) to achieve each evaluation loss for different ZO fine-tuning methods on Llama-2-7B model.



Trade-off between the accuracy and memory cost for different fine-tuning methods.

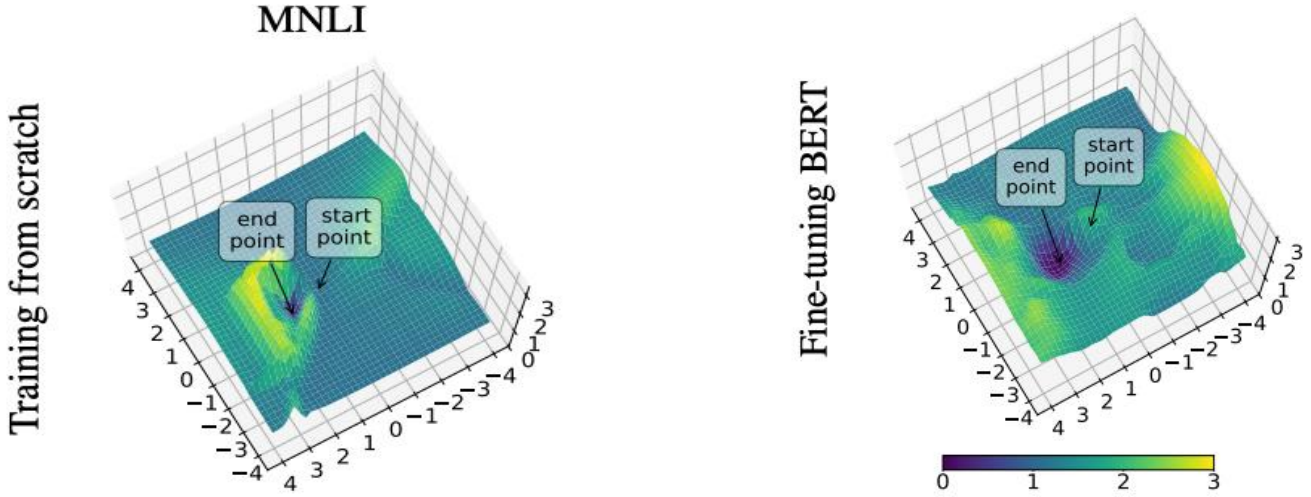
Sharpness-Aware ZO-VLM Prompt Tuning

Yifan Yang, et al, SharpZO: Hybrid Sharpness-Aware Vision Language Model Prompt Tuning via Forward-Only Passes, under review.

The Effectiveness of ZO fine-tuning: From a loss landscape perspective

The success of ZO LLM fine-tuning heavily reply on two factors:

- The loss landscape for fine-tuning is relatively **flat**, which inherently reduces the variance in ZO gradient estimation.
- The optimization starts **close** to the optimal region, meaning the parameter updates required are minimal.

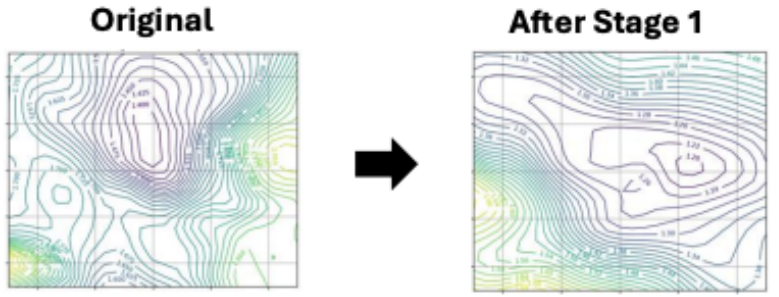


Is there a way to guide the initialization point closer to the optimal region before applying ZO optimization?

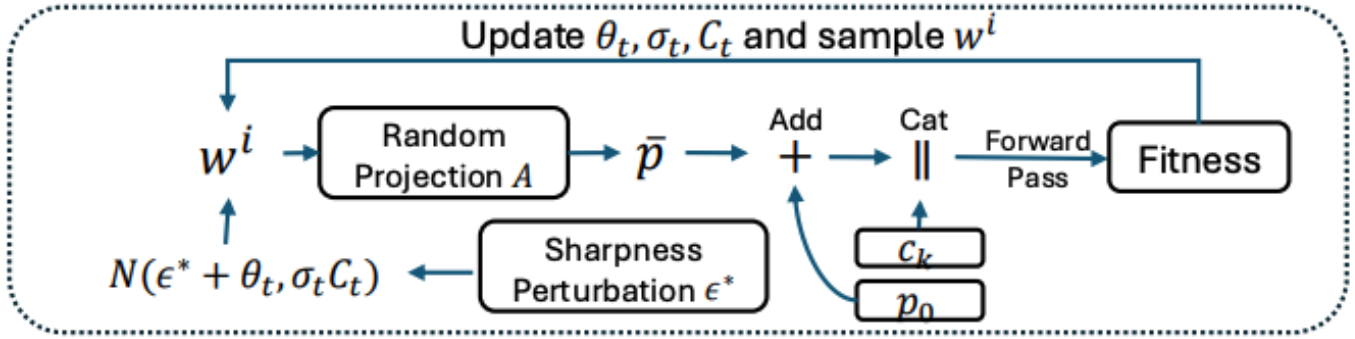
SharpZO Method

To find a better initial point for ZO optimization regarding the smoothness of loss landscape, we propose SharpZO method, which contains two stages of training:

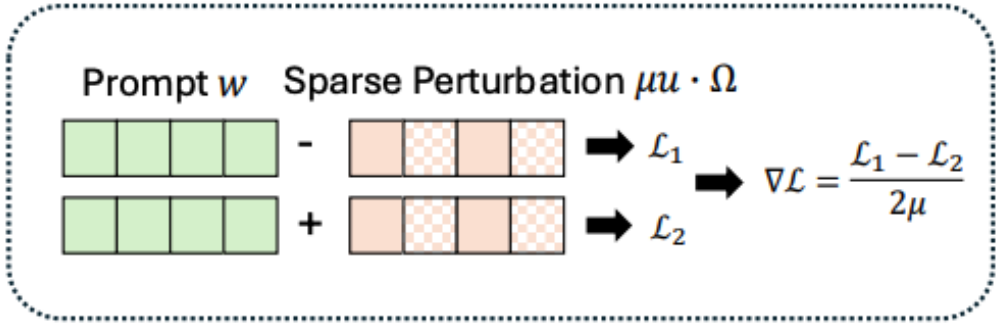
(a) SharpZO Framework Overview



(b) Evaluation on Loss landscape



(c) Stage 1: Sharpness-aware CMA-ES



(d) Stage 2: Sparse ZO optimization

SharpZO Method

To find better initial point for ZO optimization regarding the smoothness of loss landscape, we propose SharpZO method, which contains two stages of training:

	Stage 1: Evaluation Strategy	Stage 2: ZO
Exploration	Strong **global** search capability via adaptive sampling	Primarily **local** search; relies on random perturbations around current point
Computation Cost	**High** (due to population evaluations and matrix updates)	**Lower** (typically fewer perturbations; no covariance updates)

The first stage only needs to be performed around 100 steps to reduce the computation cost.

Experimental Results

Setup:

Models: CLIP w. ResNet50/Vit-B-16 backbone

Datasets: 11 classification datasets with manually initialized prompts

Training setup: Black-box prompt tuning (w. trainable prompt parameters)

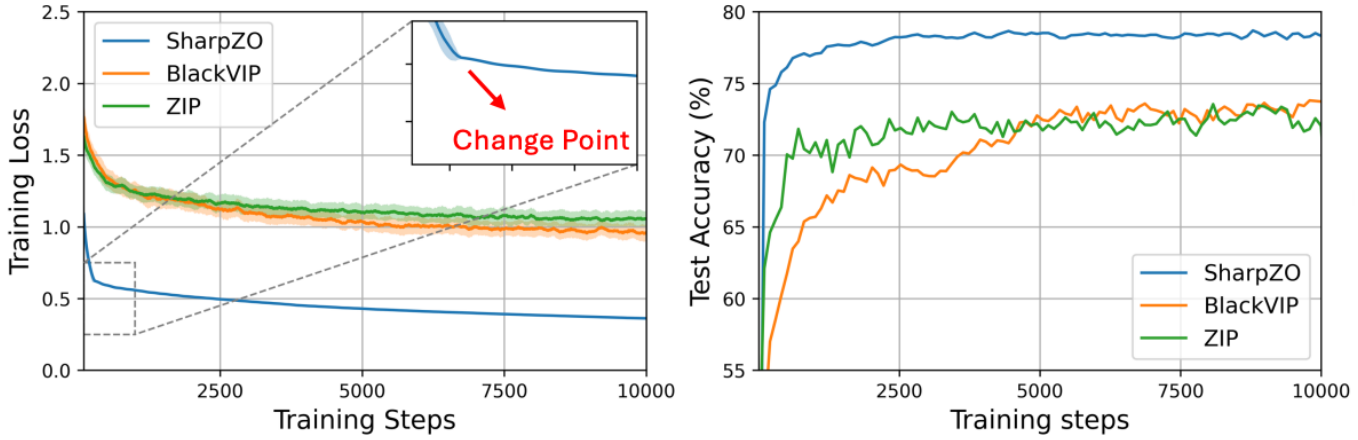
Baselines:

- BlackVIP: Naive ZO prompt tuning
- ZIP: ZO prompt tuning with low-rank prompt parameters
- Craft: CMA-ES prompt tuning combined with FO last-layer adapter tuning

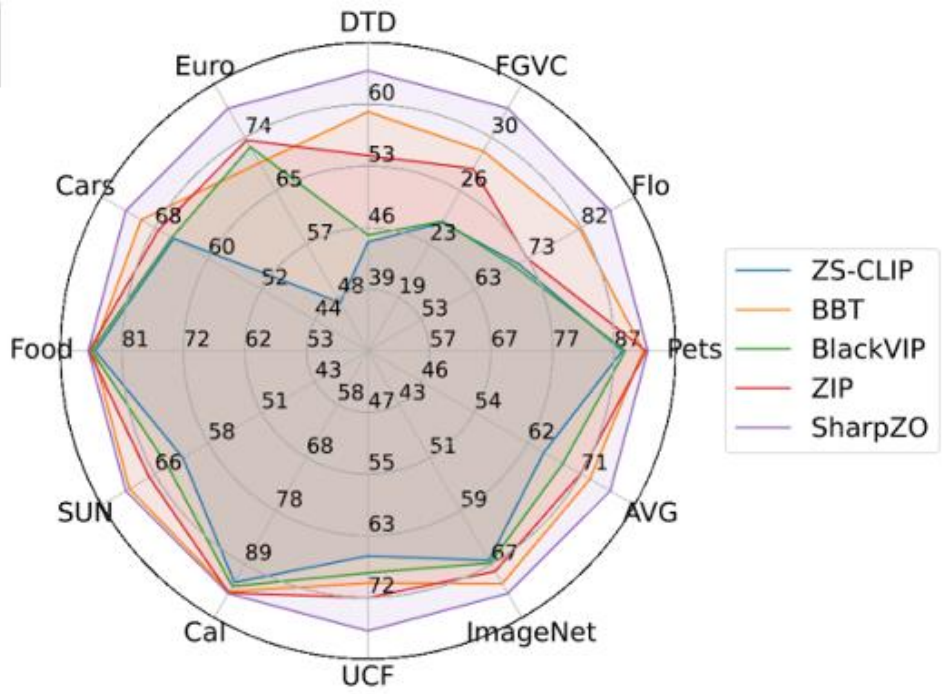
Experiential Results

- **Comparison with BP-Free Baselines**

The two-stage process of the SharpZO method **quickly** provides a strong **initialization** for the second-stage ZO training through sharpness-aware CMA-ES optimization



(a) Curve for Training Loss and Testing Accuracy on EuroSAT



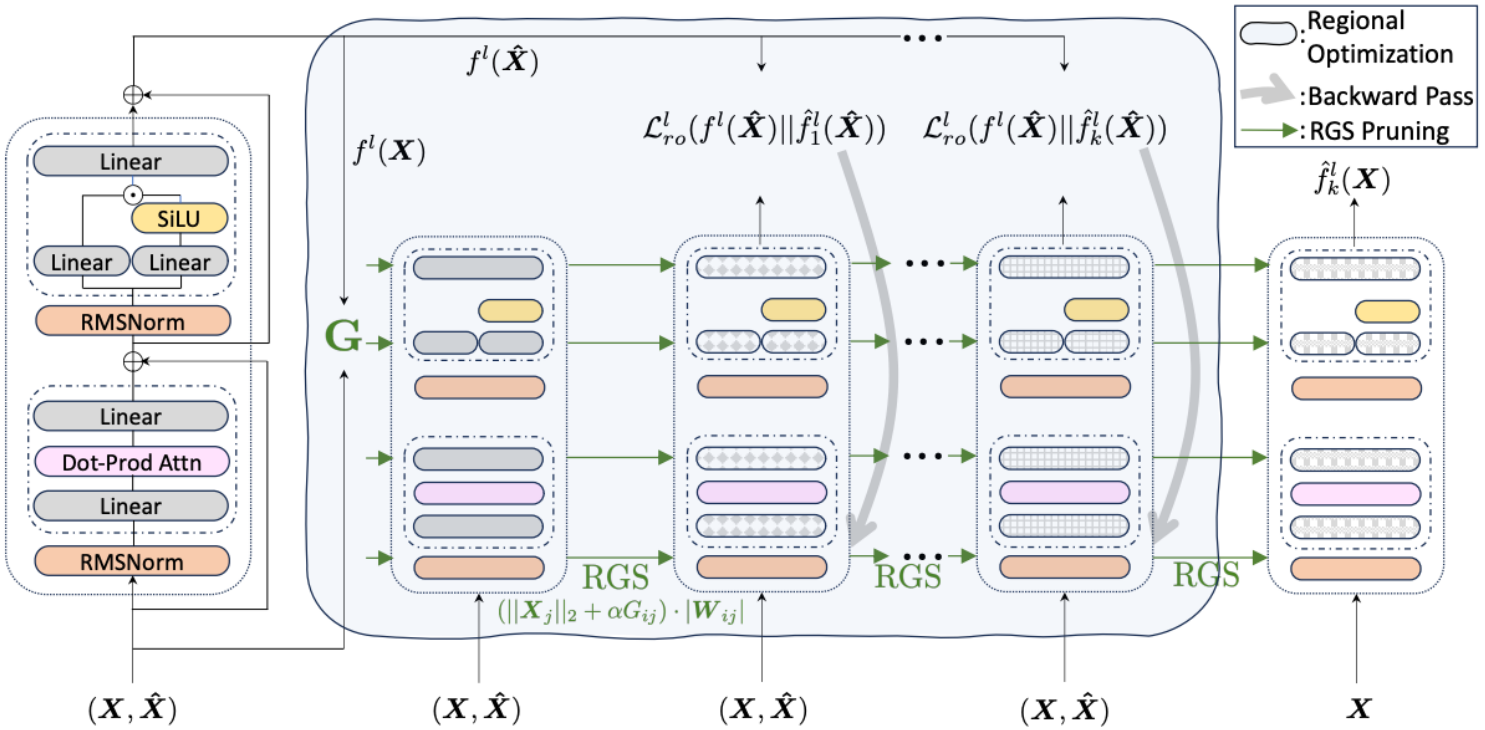
(b) Downstream Generalization

Pruning the LLMs with regional gradients

Yifan Yang et al, "Wanda++: Pruning Large Language Models via Regional Gradients", in ACL Findings 2025.

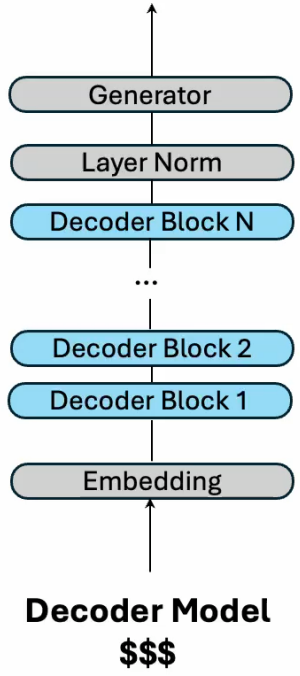
Wanda++ Methods

- Wanda++ includes a pruning score (RGS) and a regional optimization method.
  - **RGS Score**: Improve pruning metrics with regional gradients
  - **Regional Optimization**: Local update within decoder block



When to add Wanda++ as an extra spice?

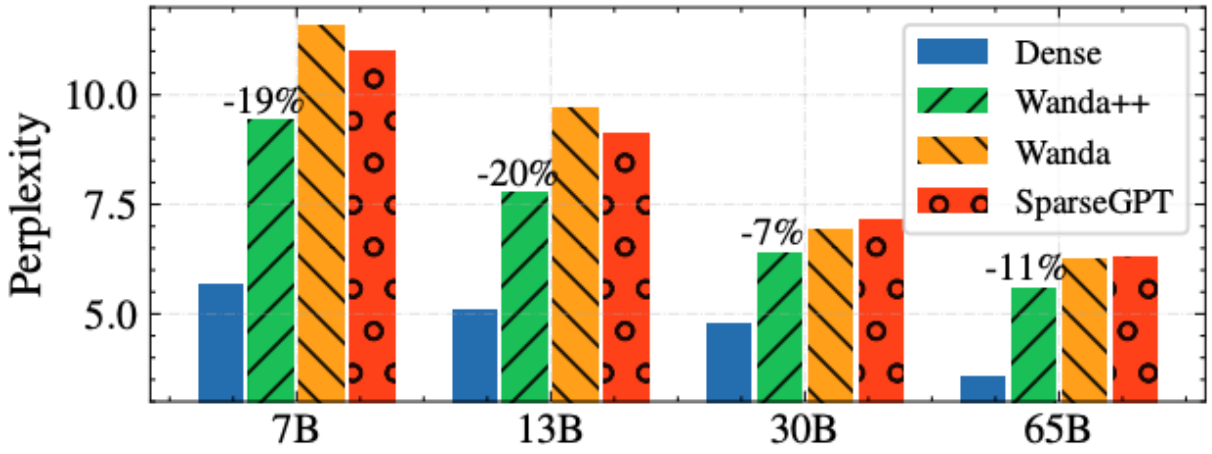
Wanda++ can be applied after post-training architectural changes (e.g., pruning, dense-to-MoE) to quickly mitigate degradation before costly recovery training.



Experiential Results

- Wikitext Perplexity

Wanda++ mitigates 2:4 pruning-induced degradation more effectively, with relative perplexity improvement over Wanda shown on Wikitext using LLaMA-1 models across four different sizes.



Experiential Results

- **Downstream Tasks**
  We compare the Zero-shot accuracy (%) from LLaMA-1 7B across various tasks under 2:4 sparsity with other baselines.

Method	Wic	Mrpc	Hellaswag	Arc_easy	Arc_challenge	Winogrande	BoolQ	RTE	MMLU
Baseline	49.84	69.12	56.96	75.29	41.80	70.00	75.02	66.43	35.10
Wanda	48.75	46.81	41.66	59.34	27.47	61.96	69.60	49.82	25.85
GBLM	49.32	65.31	41.80	61.43	30.45	63.24	**71.20**	57.43	26.34
Wanda++ RGS	49.37 (1%)	64.46 (38%)	41.43 (-1%)	62.42 (5%)	**31.06 (13%)**	62.83 (1%)	67.95 (-2%)	58.48 (17%)	26.40 (-2%)
Wanda++	**50.00 (2%)**	**68.38 (46%)**	**45.31 (8%)**	**63.72 (7%)**	29.27 (6%)	**65.04 (4%)**	67.80 (-2%)	**62.09 (24%)**	**27.52 (6%)**

- **Combined with Recovering Training**
  Wanda++ remains orthogonal to sparsity-aware fine-tuning, further reducing perplexity with LoRA to a great extent.

Methods	Dense	Pruned Model	After LoRA-tuned
Wanda	5.68	11.59	8.23(-29%)
Wanda++	5.68	9.43	6.88(-27%)

Table of Contents

Future Research

- **LLMs reasoning**
  - **How to distill the reasoning ability from large teacher models**
  - **Separate rational generation for text- and vision—dominant inputs to improve the reasoning ability of small multimodal model**
- **Zeroth-order Optimization**
  - **Use zeroth-order optimization to fine-tune the model for better reasoning ability**
  - **Use zeroth-order to do multimodal LLM fine-tuning**

Thank you and questions!
Contact me: yifanyang@ucsb.edu