Memory Efficient Low-Rank Systems for Large Vision/Language Models

Ivan Jaen-Marquez, Laik Ruetten, Sadman Sakib, Zheyang Xiong

School of Computer, Data & Information Sciences
UNIVERSITY OF WISCONSIN–MADISON

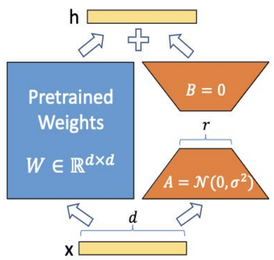
Problem Statement

Deep learning models are becoming larger, with parameter count rapidly increasing, making it more and more infeasible to train all at once due to hardware constraints on most machines.

Parameter efficient strategies allow users to train larger models with lower memory loads, allowing users to adapt these models to their needs without needing to upgrade to expensive hardware. Ideally we can get the amount of trainable parameters down enough to run on systems with lower maximum memory sizes, like consumer grade GPUs.

Related work

Low-Rank Adaptation (LoRA) was originally proposed to fine-tune by training only lower-dimensional matrices. LoRA trains less parameters and so needs less memory in training and less space for saving fine-tuned weights.



Low Rank Adaptation: Freeze W, Train A & B

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

Extensions of LoRA were proposed to utilize low-rank technique for pre-training:

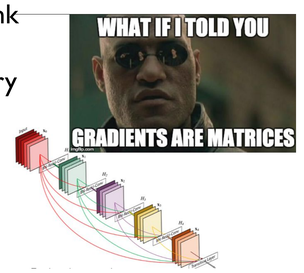
- **ReLoRA** periodically absorbs LoRA weights to main weight to gradually achieve higher ranks.
- **LTE** trains multiple LoRA heads in parallel with micro-batches and periodically merges weights.
- **GaLore** projects gradients into low-rank matrices and periodically switches projection direction.

References:

- ***[LoRA]**: Low-Rank Adaptation of Large Language Models, Hu et al. (2021)*
- ***[GaLore]**: Memory-Efficient LLM Training by Gradient Low-Rank Projection, Zhao et al. (2024)*
- ***[LTE]**: Training Neural Networks from Scratch with Parallel Low-Rank Adapters, Huh et al. (2024)*
- ***[ReLoRA]**: High-Rank Training Through Low-Rank Updates, Lialin et al. (2023)*
- ***[MeLo]**: Low-Rank Adaptation is better than fine-tuning for medical image diagnosis, Zhu et al. (2023)*
- ***[pFedLoRA]**: Model-Heterogeneous Personalized Federated Learning with LoRA Tuning, Yi et al. (2024)*

Approach and Experiments

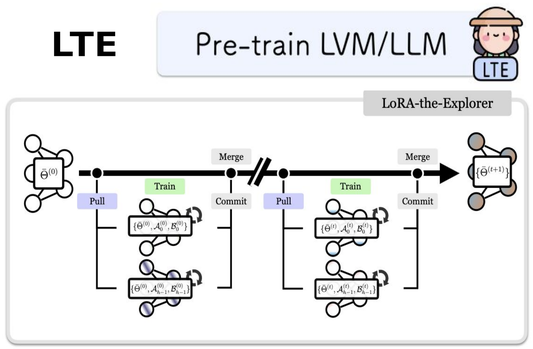
Observations and Proposal: We explore the utility of multiple low rank methods for training. We then apply these methods across task domains (vision and language). Our primary motivation is that this reduces memory costs, as this is a form of compression.



Benefits: We hope to learn which methods provide the most savings on memory load. Additionally, adapting methods to new domains proves generality and utility of these methods.

All experiments performed on NVIDIA P100 GPUs with 12 GB of memory.

LTE



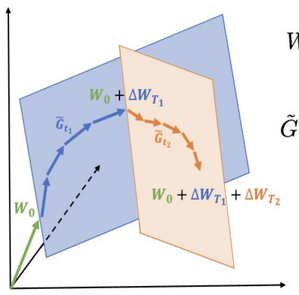
$$(1)\ \delta_{\text{lora}_n}(\mathbf{x}) = -\eta \sum_t \nabla_{\text{lora}_n}(\mathbf{x}[t])\ \ (2)\ \Delta_{\text{lora}}(\mathbf{x}) = \frac{1}{N}\sum_n \delta_{\text{lora}_n}(\mathbf{x})$$

Two techniques for training multi-head LoRA

- Multiplying each LoRA head with input and adding them up. However, it creates identical information in the LoRA heads
- Splitting input into H micro-batches, multiplying each with a LoRA head and accumulating average periodically to main weight.

GaLore

Gradient low-rank projection (GaLore) denotes the following gradient update rules (η is the learning rate):



$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \tilde{G}_t,$$

$$\tilde{G}_t = P_t \rho_t(P_t^\top G_t Q_t) Q_t^\top,$$

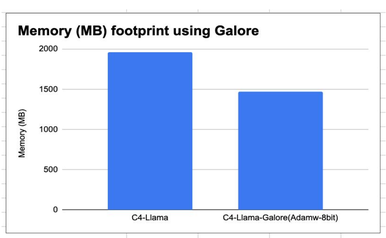
where $P_t \in \mathbb{R}^{m\times r}$ and $Q_t \in \mathbb{R}^{n\times r}$ are projection matrices.

Different from LoRA, GaLore explicitly utilizes the low-rank updates instead of introducing additional low-rank adaptors and hence does not alter the training dynamics.

Language (LLMs / GPTs)

Dataset: CNN Dailymail dataset (512 context length)
Model: GPT-2 (~124M parameters), `tiktoken` tokenizer, attention layers replaced with LTE layers

Dataset: C4 common crawl
Model: *LLaMA* 1B, hidden size: ~2K
5 training iterations
GaLore rank: 1024



Vision (LVMs / ViTs)

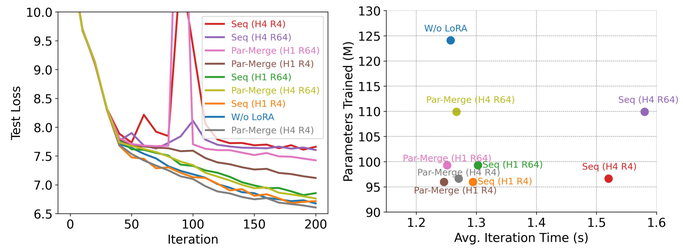
LoRA Rank	Trainable Params	Total Params	Memory Load	Accuracy	Training Time
4	0.150M	81.974M	6079MiB	0.7692200557	~23:30 min
8	0.290M	82.114M	6085MiB	0.9135097493	~23:30 min
16	0.571M	82.395M	6105MiB	0.8956824513	~23:30 min
32	1.134M	82.958M	6087MiB	0.8862116992	~23:30 min
64	2.259M	84.083M	6185MiB	0.8440111421	~23:30 min
Full Fine Tuning	82.557M	82.557M	9225MiB	0.5603064067	~32:00 min

Training for 1 epoch; Recording training curves is a work in progress.

Dataset: MedMNIST/PathMNIST; image size: 224
Model: `vit_base_patch16_224` with normal LoRA (varied ranks)
Optimizer: `GaLoreAdamW` (rank 512)

Varying *GaLore* rank has little affect `vit_large_patch16_224` has same trends.
Adam vs *GaLore* has similar memory trends as *C4-LlaMA* above, but not as significant.

Analysis



LTE (low rank weights):

- **Parallel-Merge**: Performs better than sequential multi-head training and close to full parameter training.
- **Multi-Head**: Better performance with more compute
- **Memory Efficiency**: 24% less memory for parameters with similar accuracy and training time

GaLore (low-rank gradients):

GaLore's authors claim running on a "consumer-grade" (still top tier and expensive) RTX 4090 GPU with 25GB of memory, but we dispute GaLore's effectiveness across scale. In our experience, we trained with 12GB of memory and found that GaLore's benefits were minimal for the ViTs because of their lower param count (~82M for base, ~220M for large, ~630M for huge) compared to language models like Llama (1B params, and goes bigger). GaLore is only substantial for these models with billions of parameters and GPUs with significant memory already.

Future Work

- Combine the complementary benefits of both LTE and GaLore to further improve training performance.
- Perform deeper studies to derive heuristics for setting hyperparameters (rank, etc) for a given task.
- An example application of LoRAs is in hospital systems. Adaptors could be multilayered, with a medical foundation model for different diagnosis tasks that can be adapted off of a regular ViT pre-trained on natural images ImageNet [MeLo]. Federated learning can protect data privacy, maintain adaptability, while still contributing to model performance [pFedLoRA].