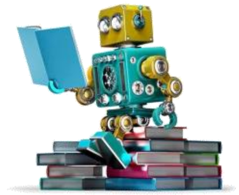
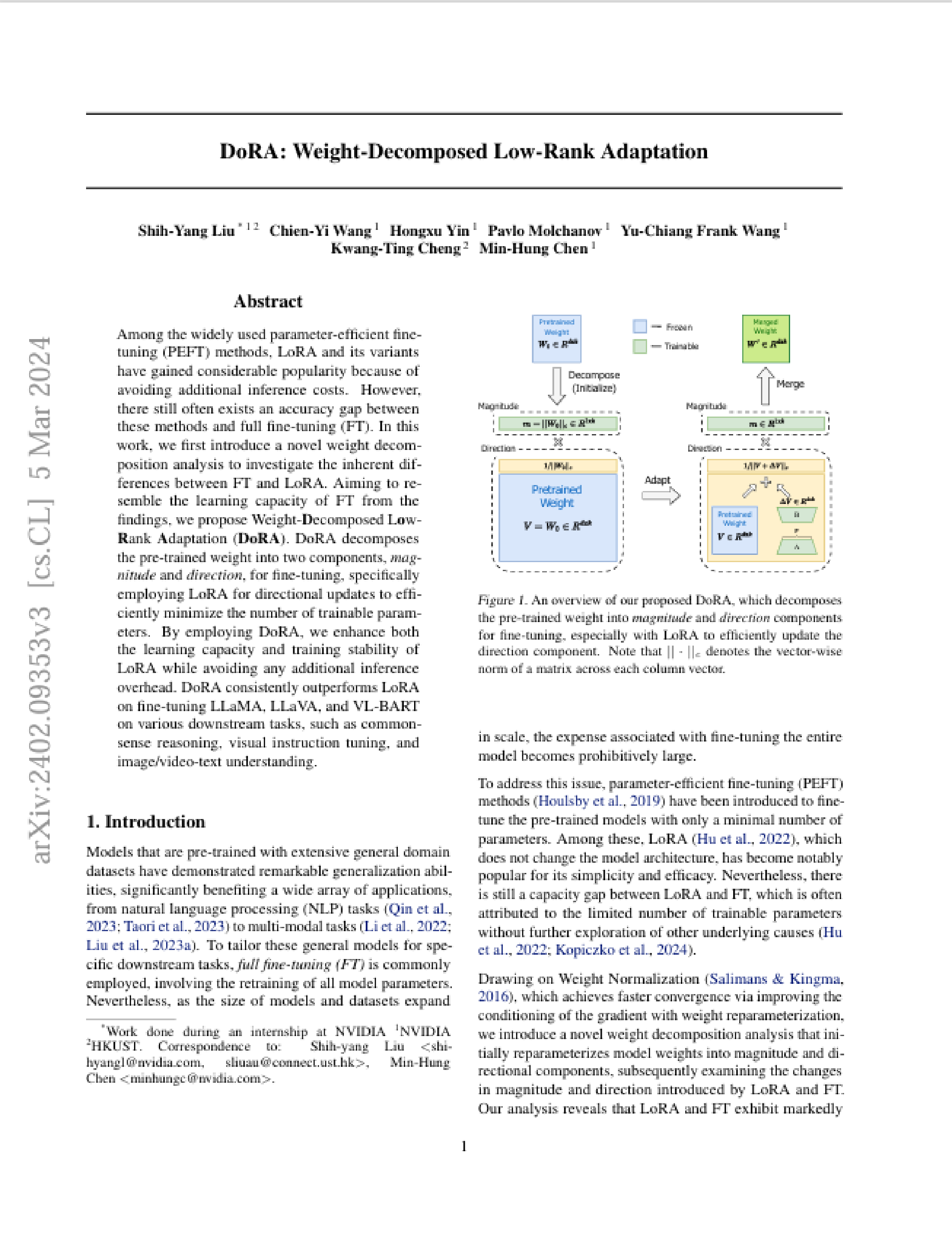
SDML
ML Paper Review

April 2024



DoRA: Weight-Decomposed Low-Rank Adaptation

https://arxiv.org/abs/2402.09353



arXiv:2402.09353v3 [cs.CL] 5 Mar 2024

DoRA: Weight-Decomposed Low-Rank Adaptation

Shih-Yang Liu [*,1,2] Chien-Yi Wang [1] Hongxu Yin [1] Pavlo Molchanov [1] Yu-Chiang Frank Wang [1]
Kwang-Ting Cheng [2] Min-Hung Chen [1]

Abstract

Among the widely used parameter-efficient fine-tuning (PEFT) methods, LoRA and its variants have gained considerable popularity because of avoiding additional inference costs. However, there still often exists an accuracy gap between these methods and full fine-tuning (FT). In this work, we first introduce a novel weight decomposition analysis to investigate the inherent differences between FT and LoRA. Aiming to resemble the learning capacity of FT from the findings, we propose **W**eight-**D**ecomposed Low-**R**ank **A**daptation (**DoRA**). DoRA decomposes the pre-trained weight into two components, *magnitude* and *direction*, for fine-tuning, specifically employing LoRA for directional updates to efficiently minimize the number of trainable parameters. By employing DoRA, we enhance both the learning capacity and training stability of LoRA while avoiding any additional inference overhead. DoRA consistently outperforms LoRA on fine-tuning LLaMA, LLaVA, and VL-BART on various downstream tasks, such as commonsense reasoning, visual instruction tuning, and image/video-text understanding.

Figure 1. An overview of our proposed DoRA, which decomposes the pre-trained weight into *magnitude* and *direction* components for fine-tuning, especially with LoRA to efficiently update the direction component. Note that $\|\cdot\|_c$ denotes the vector-wise norm of a matrix across each column vector.

1. Introduction

Models that are pre-trained with extensive general domain datasets have demonstrated remarkable generalization abilities, significantly benefiting a wide array of applications, from natural language processing (NLP) tasks (Qin et al., 2023; Taori et al., 2023) to multi-modal tasks (Li et al., 2022; Liu et al., 2023a). To tailor these general models for specific downstream tasks, *full fine-tuning (FT)* is commonly employed, involving the retraining of all model parameters. Nevertheless, as the size of models and datasets expand in scale, the expense associated with fine-tuning the entire model becomes prohibitively large.

To address this issue, parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019) have been introduced to fine-tune the pre-trained models with only a minimal number of parameters. Among these, LoRA (Hu et al., 2022), which does not change the model architecture, has become notably popular for its simplicity and efficacy. Nevertheless, there is still a capacity gap between LoRA and FT, which is often attributed to the limited number of trainable parameters without further exploration of other underlying causes (Hu et al., 2022; Kopiczko et al., 2024).

Drawing on Weight Normalization (Salimans & Kingma, 2016), which achieves faster convergence via improving the conditioning of the gradient with weight reparameterization, we introduce a novel weight decomposition analysis that initially reparameterizes model weights into magnitude and directional components, subsequently examining the changes in magnitude and direction introduced by LoRA and FT. Our analysis reveals that LoRA and FT exhibit markedly

[*]Work done during an internship at NVIDIA [1]NVIDIA [2]HKUST. Correspondence to: Shih-yang Liu <shihyangl@nvidia.com, sliuau@connect.ust.hk>, Min-Hung Chen <minhungc@nvidia.com>.

DoRA overview

- DoRA: Weight-Decomposed Low-Rank Adaptation
- Shih-Yang Liu et al. mostly from NVIDIA (2024)

- Large foundation models pre-trained on general tasks work well for a wide variety of downstream tasks
- Fine-tuning these models can improve downstream task performance, but full fine-tuning is expensive
- DoRA is one of many techniques for *parameter efficient fine-tuning*, which requires less memory and is faster

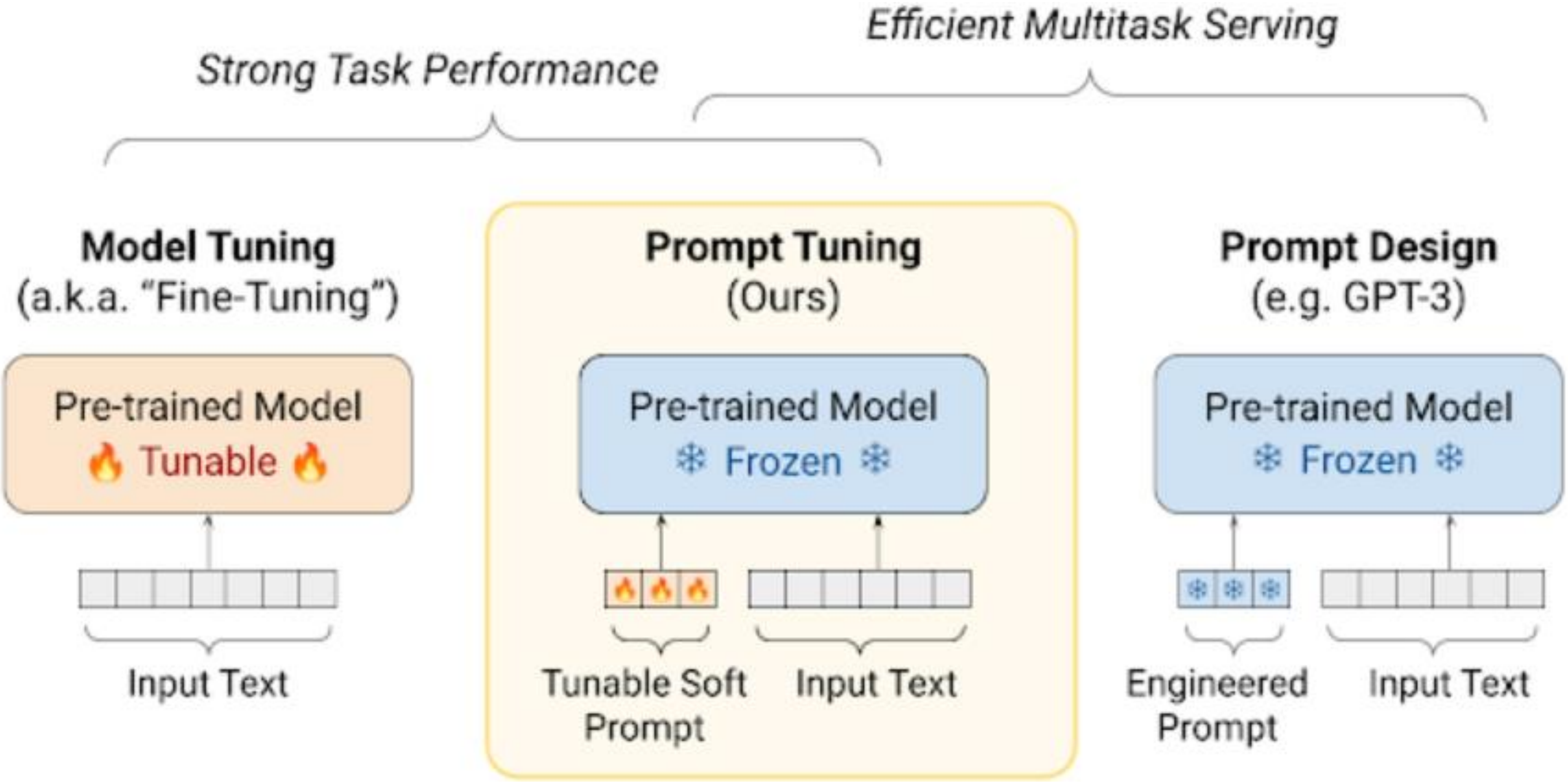
Parameter Efficient Fine-Tuning

- Goal of parameter efficient fine-tuning (PEFT) is to get as much of the benefits of full fine-tuning with the least cost
- For LLMs prompt engineering designs hard (text) prompts
  - Takes advantage of in-context learning capability of LLMs
  - It doesn't require any additional training but is hard to optimize, limited in benefit, and uses up tokens
- Two main approaches to PEFT
  - Soft prompts train a small number of token positions prior to the text
  - Adapters add small components to the network and train them while keeping the rest of the model weights frozen

Prompt tuning [1]

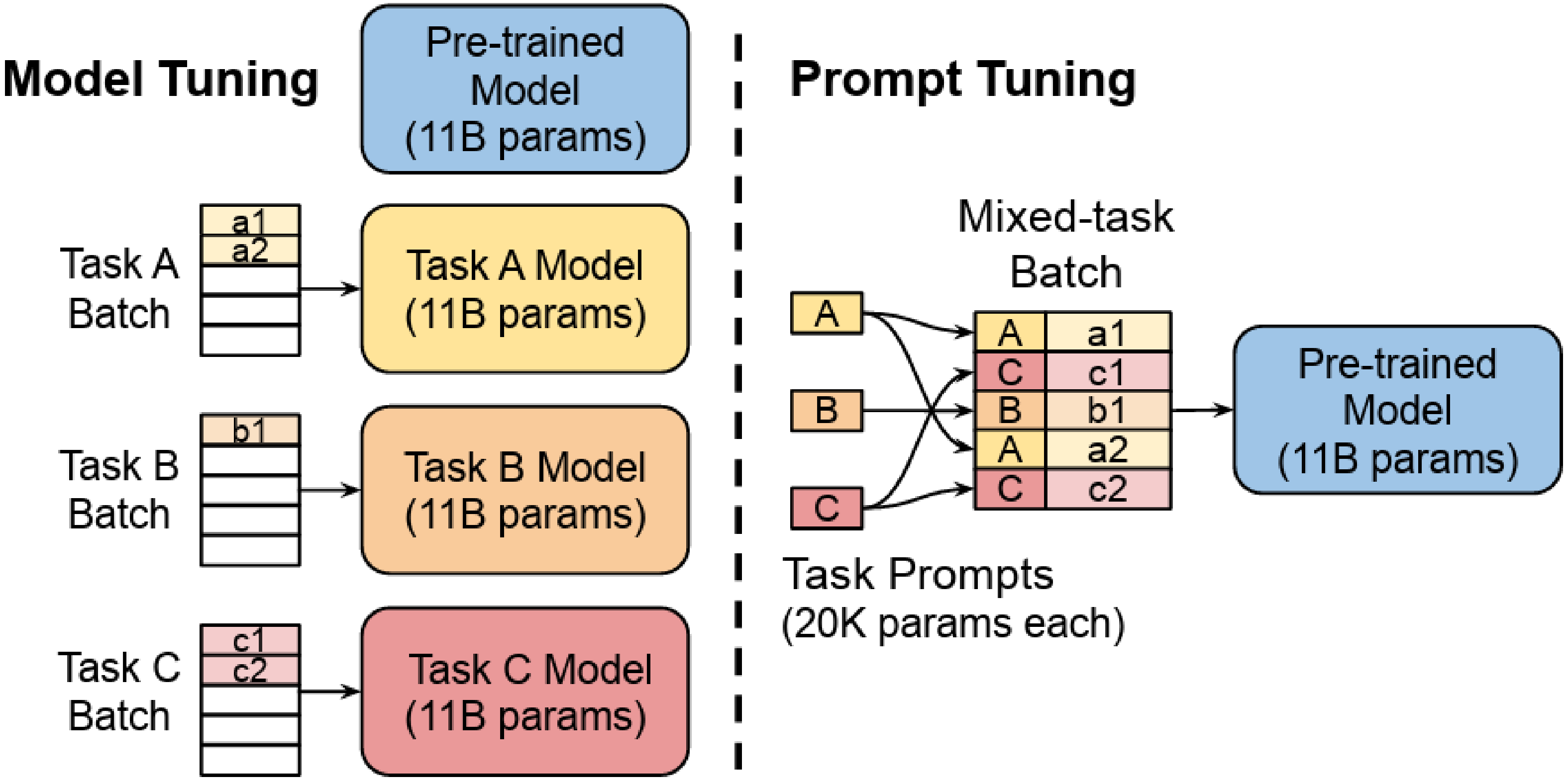
- Lester et al. (2021) is often cited for the approach to training soft prompts without modifying model weights
- Dense embedding vectors are trained for a handful of token positions
- To train the soft prompt:
  - Use prompt and completion pairs, with cross-entropy loss
  - Freeze all of the base model weights
  - Only backpropagate the embedding vectors
- Once a prompt is trained, it is pre-pended before the text tokens

Prompt tuning [2]



Prompt tuning [3]

- With prompt tuning, multiple soft prompts can be created, and they can be mixed within batches:

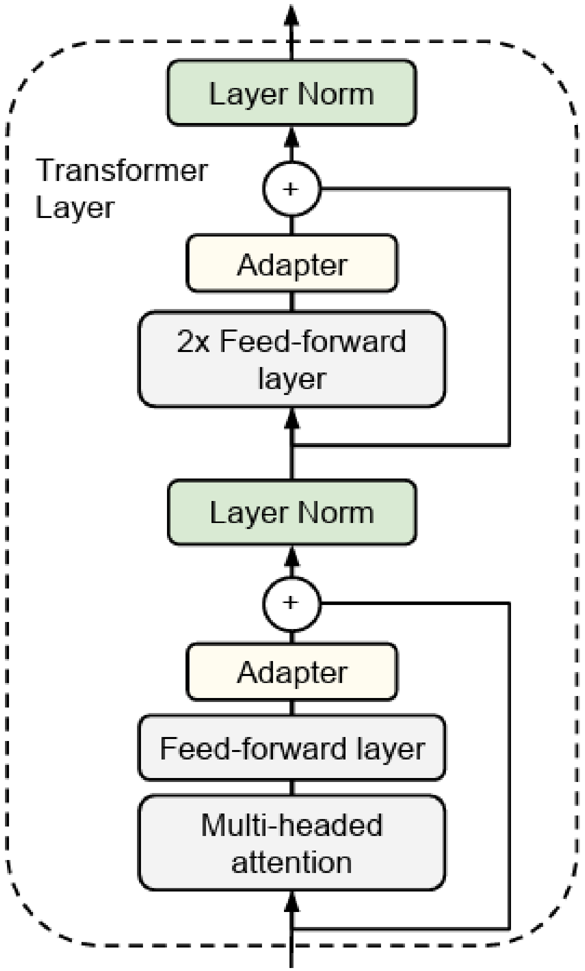


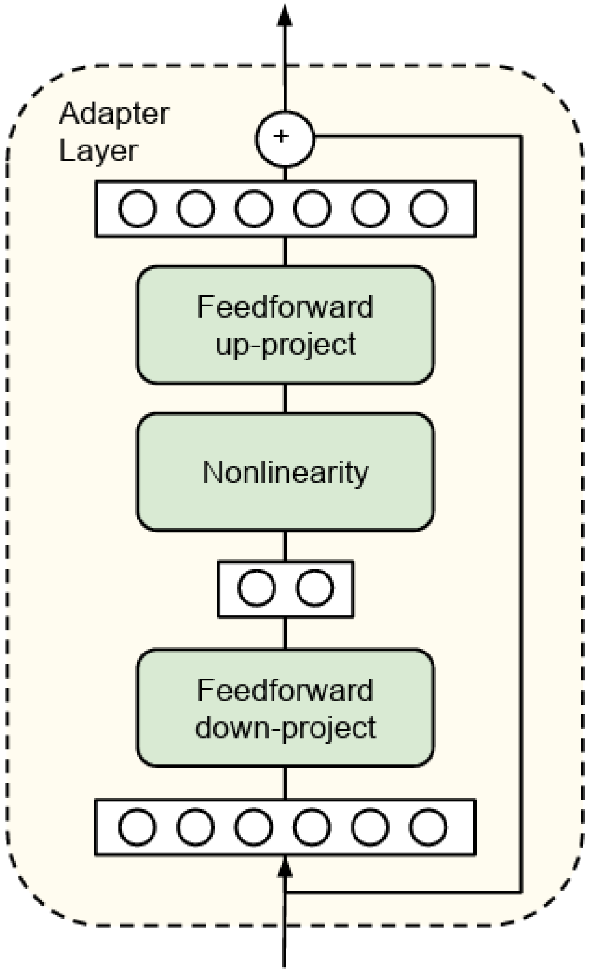
Other soft prompt techniques

- There several other well-known variations on the soft prompt theme
- Prefix tuning not only trains a handful of vectors at the embedding layer, but also vectors at every layer of the model
- P-tuning adds additional logic (a LSTM) to decide between different soft prompt choices

Adapters

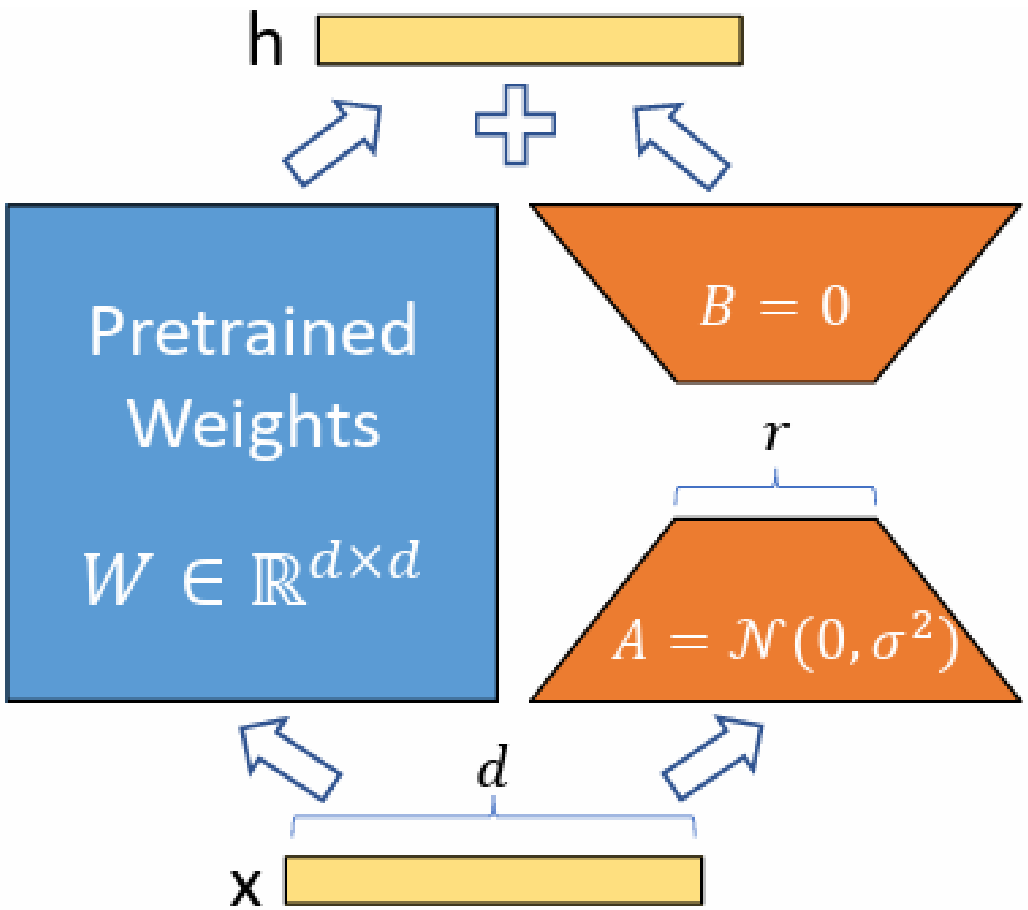
- Houlsby et al. (2019) introduced adapters
- Add small components to the model
- To train, freeze the original weights and only train the adapters
- Adds inference cost
  - Adapters require more memory
  - Worse GPU pipelining





LoRA [1]

- Hu et al. (2021) introduced a new adapter that was separate during training, but could be combined with the original weights during inference
- Multiplying two low rank matrices (*d* x *r* and *r* x *d*) is cheaper but results in *d* x *d*
- Cost-performance tradeoff can be adjusted by varying rank *r*



LoRA [2]

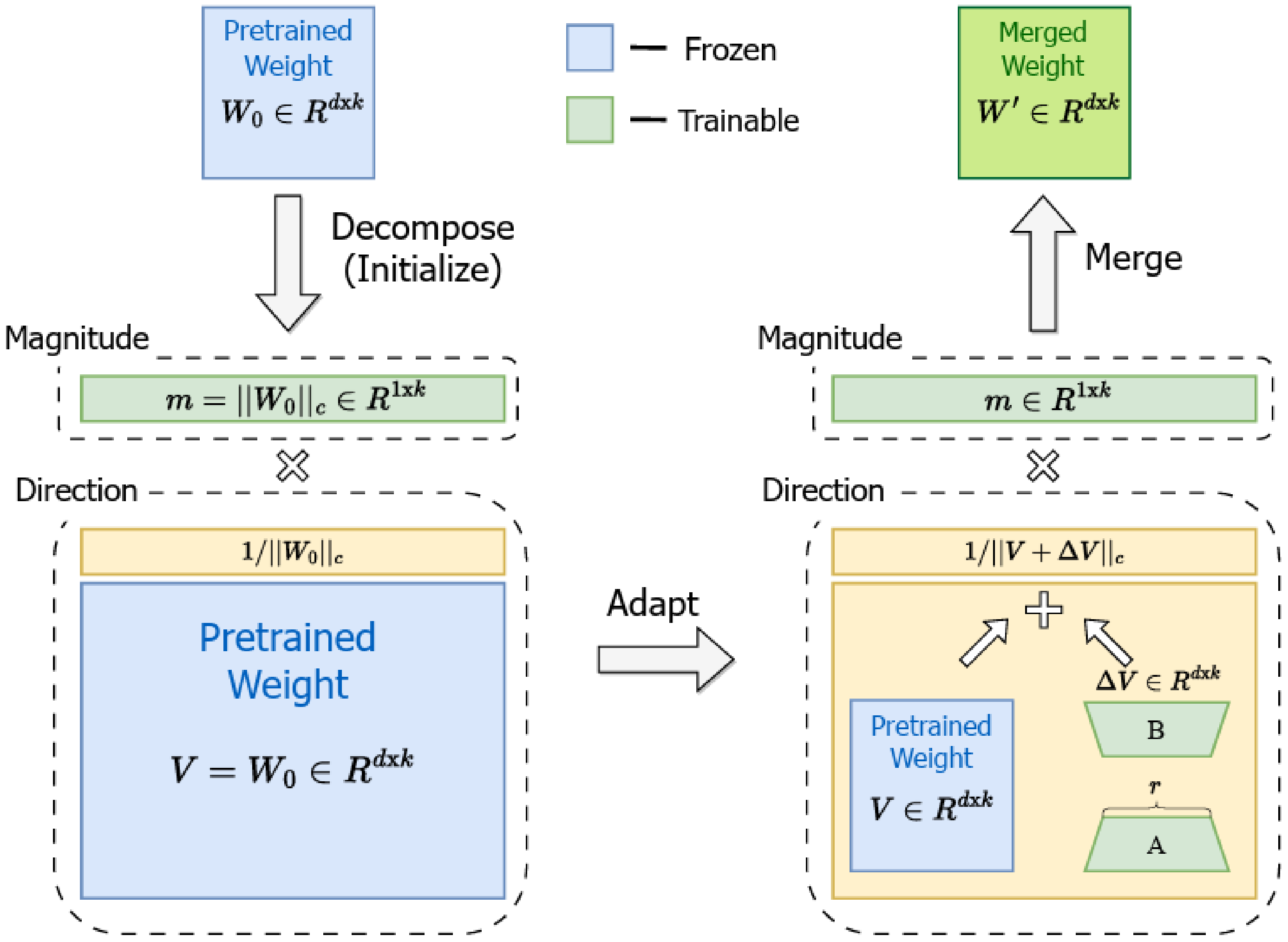
- For training, the product of low rank matrices sits next to the regular weight matrix
  - Its product is added to the weight matrix, and that sum is used normally
- After training, one option is to permanently add the adapter product to the original weights, creating a new set of weights
  - These new weights follow the original architecture, adding zero inference cost
  - (Another possibility is to rapidly switch weights on the fly by adding in different small-sized adapter weights for different tasks)
- You can choose which subset of weights you want to add LoRA
- The success of LoRA has spurred countless variants, including QLoRA, LoHa, LoKr, and VeRA

DoRA concept [1]

- DoRA differs from LoRA in that the original model weights are decomposed into two components prior to applying LoRA
- Each weight matrix is split into a magnitude vector and a direction matrix
  - The direction matrix is created by scaling each column to be unit length
  - The magnitude vector stores the scaling factors needed for each column
- For training:
  - The magnitude vectors are small, and are trained normally
  - The direction matrices are big, so are frozen and LoRA is used
- For inference, the magnitude vectors and updated direction matrices can be combined back into updated weights for the original model

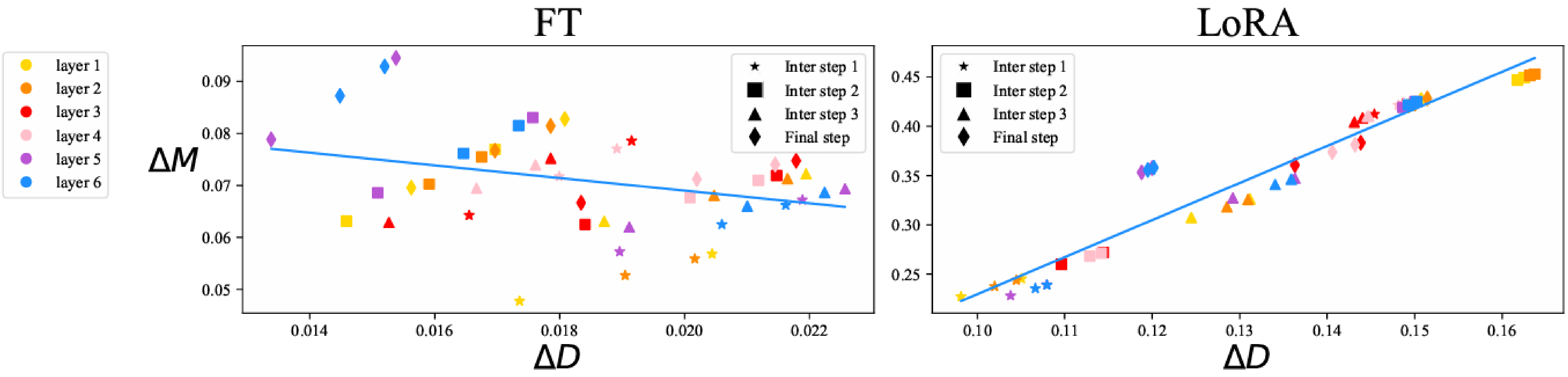
DoRA concept [2]

- Upper left is original weight matrix
- Lower left is the two part decomposition of magnitude and direction
- Lower right freezes direction matrix and uses LoRA for training
- Upper right reconstructs original weight matrix shape with new values



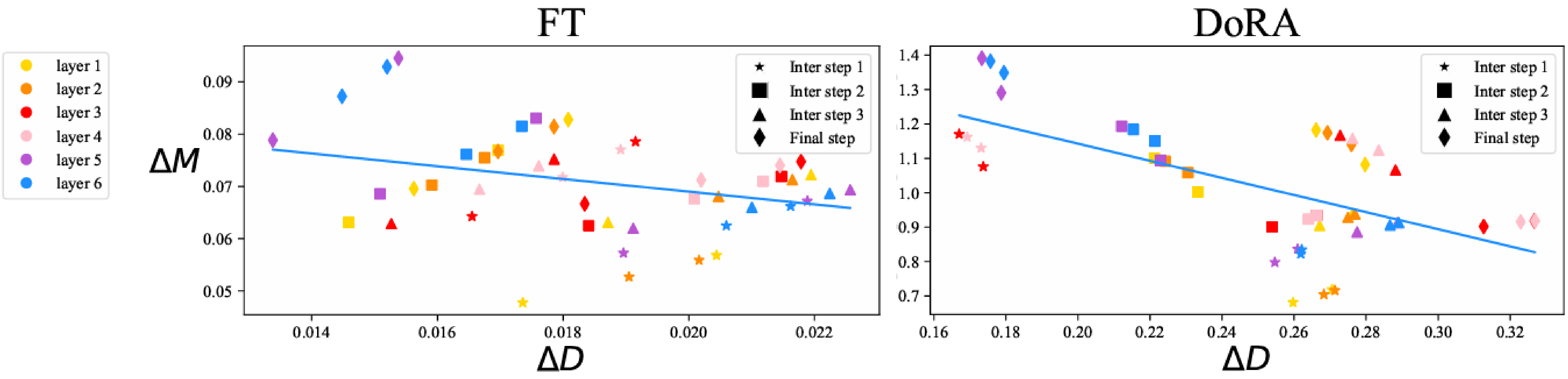
DoRA motivation [1]

- Authors found that changes to magnitude and direction for full fine-tuning are largely independent (small negative correlation)
- But LoRA creates highly correlated changes, hurting performance



DoRA motivation [2]

- DoRA fixes the problem, resulting in pattern more similar to full fine-tuning, improving performance compared to LoRA



DoRA results [1]

- First test was commonsense reasoning with LLaMA language models

Table 1. Accuracy comparison of LLaMA 7B/13B with various PEFT methods on eight commonsense reasoning datasets. Results of all the baseline methods are taken from (Hu et al., 2023). DoRA†: the adjusted version of DoRA with the rank halved.

Model	PEFT Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA	0.83	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA† (Ours)	0.43	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	**77.5**
	DoRA (Ours)	0.84	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	**78.1**
LLaMA-13B	Prefix	0.03	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series	0.80	71.8	83	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel	2.89	72.5	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.4
	LoRA	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA† (Ours)	0.35	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	**80.8**
	DoRA (Ours)	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	**81.5**

DoRA results [2]

- Also beat LoRA on multimodal tasks with images & video on VL-BART, and with visual instruction tuning on LLaVA

Table 2. The multi-task evaluation results on VQA, GQA, NVLR^2 and COCO Caption with the VL-BART backbone.

Method	# Params (%)	VQA^{v2}	GQA	NVLR^2	COCO Cap	Avg.
FT	100	66.9	56.7	73.7	112.0	77.3
LoRA	5.93	65.2	53.6	71.9	115.3	76.5
DoRA (Ours)	5.96	65.8	54.7	73.1	115.9	**77.4**

Table 3. The multi-task evaluation results on TVQA, How2QA, TVC, and YC2C with the VL-BART backbone.

Method	# Params (%)	TVQA	How2QA	TVC	YC2C	Avg.
FT	100	76.3	73.9	45.7	154	87.5
LoRA	5.17	75.5	72.9	44.6	140.9	83.5
DoRA (Ours)	5.19	76.3	74.1	45.8	145.4	**85.4**

Table 4. Visual instruction tuning evaluation results for LLaVA-1.5-7B on a wide range of seven vision-language tasks. We directly use checkpoints from (Liu et al., 2023a) to reproduce their results.

Method	# Params(%)	Avg.
FT	100	66.5
LoRA	4.61	66.9
DoRA (Ours)	4.63	**67.6**

DoRA results [3]

- Tried comparing to VeRA, then combining with it
- Also confirmed with limited amount of instruction tuning data

Table 5. Average scores on MT-Bench assigned by GPT-4 to the answers generated by fine-tuned LLaMA-7B/LLaMA2-7B.

Model	PEFT Method	# Params (%)	Score
LLaMA-7B	LoRA	2.31	5.1
	DoRA (Ours)	2.33	**5.5**
	VeRA	0.02	4.3
	DVoRA (Ours)	0.04	**5.0**
LLaMA2-7B	LoRA	2.31	5.7
	DoRA (Ours)	2.33	**6.0**
	VeRA	0.02	5.5
	DVoRA (Ours)	0.04	**6.0**

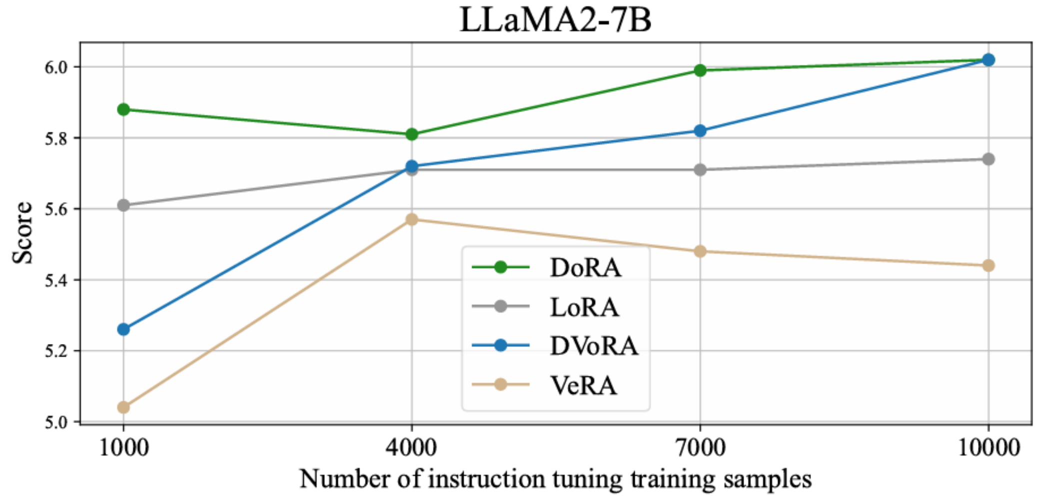


Figure 3. Performance of fine-tuned LLaMA2-7B on MT-Bench using different numbers of Alpaca training samples.

DoRA ablations

- Tested different rank settings
- Also ablated using just the magnitude adapter in some components

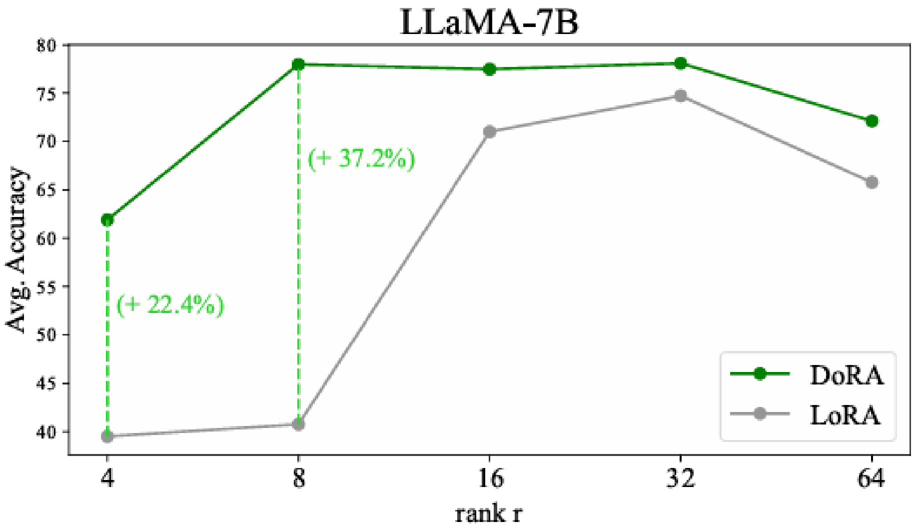


Figure 4. Average accuracy of LoRA and DoRA for varying ranks for LLaMA-7B on the commonsense reasoning tasks.

Table 6. Accuracy comparison of LLaMA 7B/13B with two different tuning granularity of DoRA. Columns **m** and **V** designate the modules with tunable magnitude and directional components, respectively. Each module is represented by its first letter as follows: (Q)uery, (K)ey, (V)alue, (O)utput, (G)ate, (U)p, (D)own.

Model	PEFT Method	# Params (%)	m	V	Avg.
	LoRA	0.83	-	-	74.7
LLaMA-7B	DoRA (Ours)	0.84	QKVUD	QKVUD	78.1
	DoRA (Ours)	0.39	QKVOGUD	QKV	77.5
	LoRA	0.67	-	-	80.5
LLaMA-13B	DoRA (Ours)	0.68	QKVUD	QKVUD	81.5
	DoRA (Ours)	0.31	QKVOGUD	QKV	81.3

DoRA conclusion

- Authors found that LoRA trains such that it artificially forces large changes in direction to have large changes in magnitude & vice versa
- By decomposing each weight matrix into a magnitude vector and a direction matrix, they decoupled magnitude and direction changes
- Consistently beat LoRA performance with many different models, many different tasks, and for range of rank settings
- Also improved VeRA, so might generalize to other variants too
- When fine-tuning LLMs, not only can you choose which parts of the model to apply the full DoRA training, you can also opt to train some parts on just the cheaper magnitude training

References [1]

- Parameter-Efficient Transfer Learning for NLP
  Neil Houlsby et al. (2019)
  https://arxiv.org/abs/1902.00751
- The Power of Scale for Parameter-Efficient Prompt Tuning
  Brian Lester et al. (2021)
  https://arxiv.org/abs/2104.08691
- LoRA: Low-Rank Adaptation of Large Language Models
  Edward J. Hu et al. (2021)
  https://arxiv.org/abs/2106.09685
- Hugging Face PEFT documentation
  https://huggingface.co/docs/peft/index

References [2]

- VeRA: Vector-based Random Matrix Adaptation
  Dawid J. Kopiczko et al. (2023)
  https://arxiv.org/abs/2310.11454