LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen

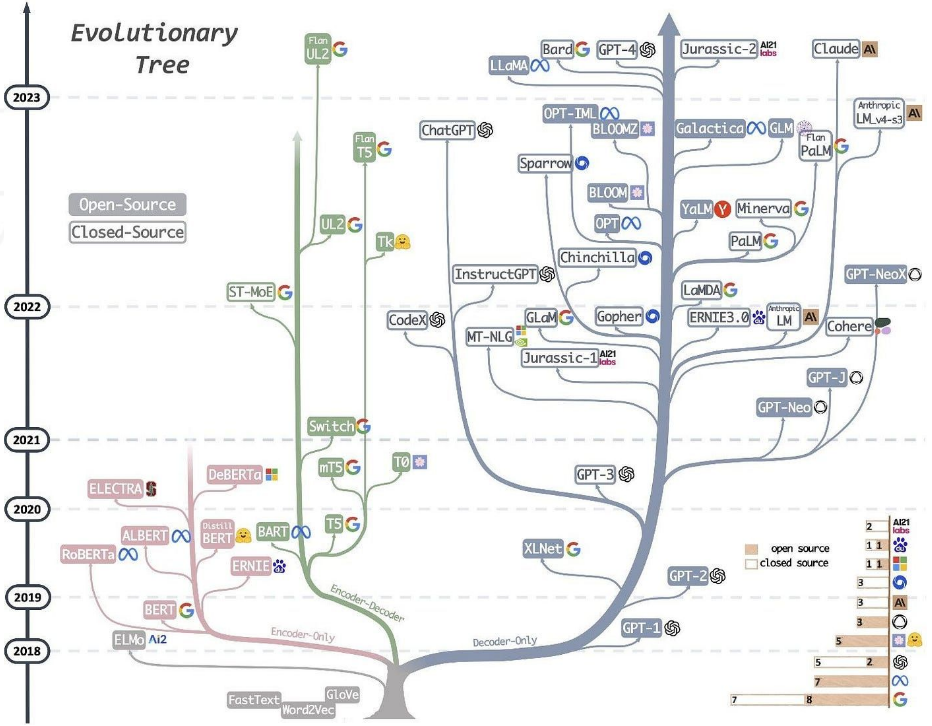
Presented by: Kien Tran, Ashish Dhiman

Agenda

1. Problem introduction
2. Background
3. Method
4. Experimental Analysis
5. Strengths and Weaknesses
6. Further developments

Problem introduction

Development of Large Language Models (LLMs)



Yang et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, Arvix 2023

Problems with Foundation LLMs

- General-purpose - i.e. not task-specific
- VERY LARGE Models

```
pip install -q meaningcloud-python

§ Output

> stdout : ['Note: you may need to restart the kernel to use updated packages.\n']


§ Code

import meaningcloud
#Dictionary to store API Keys
DC=
{'AS':meaningcloud.MeaningCloud('YOUR_LICENSE_KEY'),'DV':meaningcloud.MeaningCloud('YOUR_OTHE
R_LICENSE_KEY')}
#Automatic categorization

categorization_req = {
    'key': DC['DV'].user_key,
    'lang': 'en',
    'url': 'http://www.dcc.uchile.cl/'
  }
```

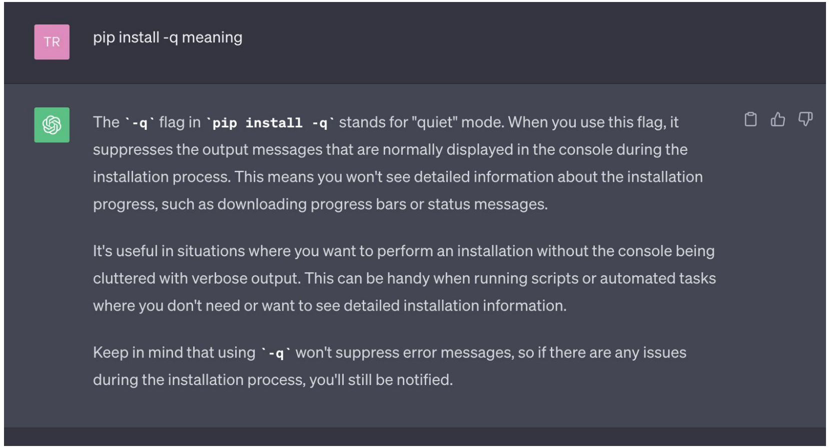
Georgia Tech

Domain Adaptation

Domain Adaptation is a type of **transfer learning** that involves training a model with **data from a source domain**.

Examples:

Chat Conversation



Code Generation

Georgia Tech

Problem Statement: Fine-tuning LLMs

Base model	**Full Fine-tuning (Adaptation)**	**Parameter-Efficient Adaptation**
Parameters		
Φ_0	$\Phi_0 + \Delta\Phi$	$\Phi_0 + \Delta\Phi(\Theta)$
GPT4, LLaMA, etc.	GPT-3 with $\lvert\Phi_0\rvert \approx \boxed{175 \text{ Billion} \approx \lvert\Delta\Phi\rvert}$	$\lvert\Theta\rvert \ll \lvert\Phi_0\rvert$
	- Large hardware requirements *- Costly training, storage, and inference*	
Adapting Ojective		
	$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{\lvert y\rvert} \log\left(P_{\Phi}(y_t \mid x, y_{<t})\right)$	$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{\lvert y\rvert} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t \mid x, y_{<t})\right)$
	Adapting to domain-specific dataset Z	

Georgia Tech

Background

Existing Methods & Limitations (1/2)

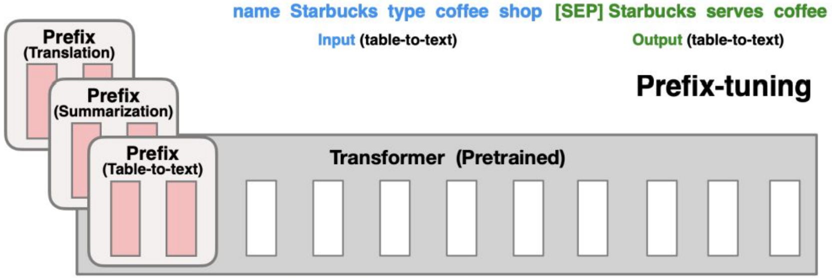
Methods

Prompt Engineering

Describe the **instructions** and **examples** of the task (one-shot, few-shots learning) **with words**

Continuous prompts (e.g. Li & Liang, 2021)

Instead of discrete prompts (words), use continuous prompts (trainable special vectors)



Limitations

- Unreliable performance ("prompt engineering is an art")
- Waste computing power processing prompts

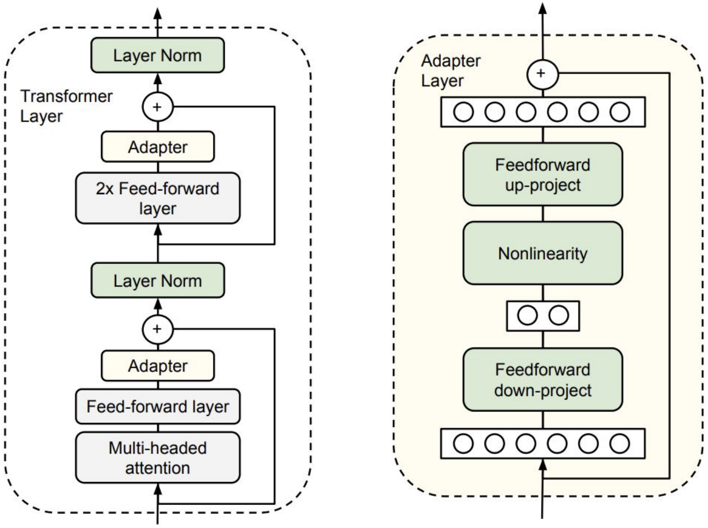
- Valuable token space must be spent on prefix token embeddings
- Increase inference time
- Not so sure regarding scalability

Existing Methods & Limitations (2/2)

Methods

Adapter-based (e.g. Houlsby et al., 2019)

Insert low-rank, trainable adapter layers between existing layers



Limitations

- Multi-headed attention weights was not changed
- Make the model deeper, thus introduce additional latency during inference
- Not able to out-perform full fine-tuning baseline

Method

LoRA

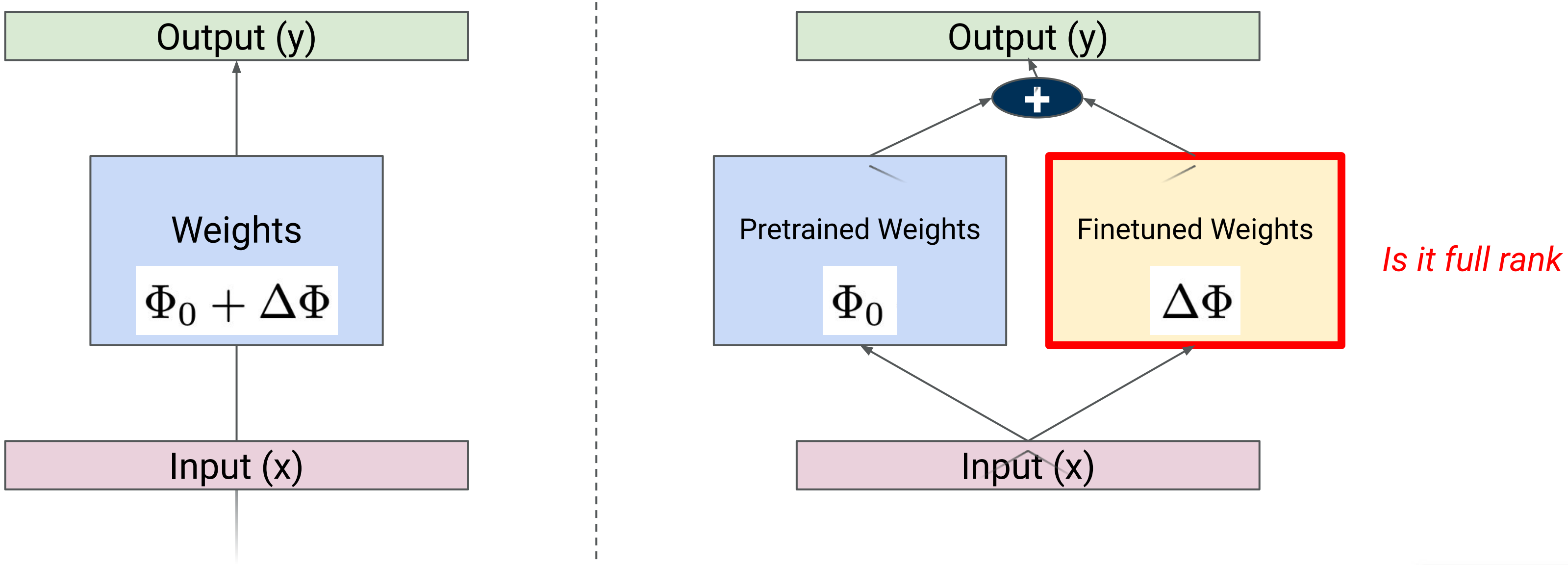
Why do we need it ?

- It's too expensive to fine-tune all parameters in a large model.

$$\boxed{\Phi'} = \boxed{\Phi_0} + \boxed{\Delta\Phi}$$

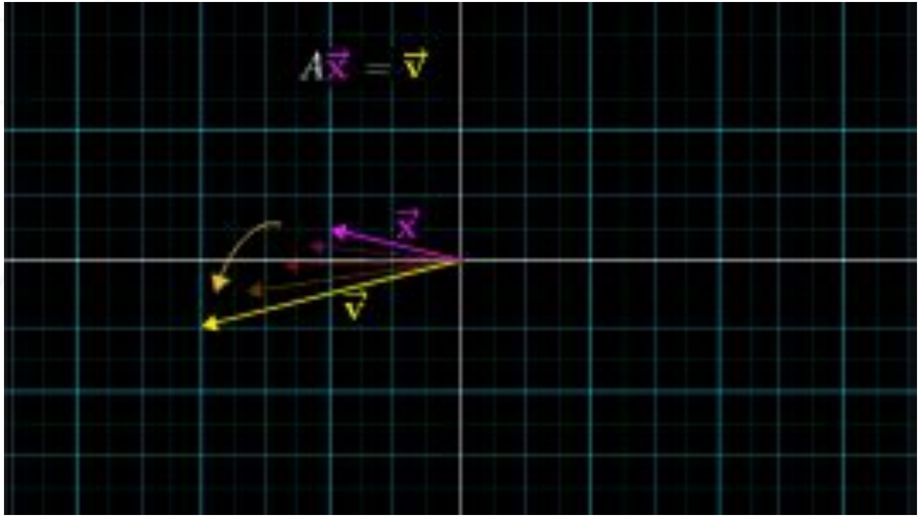
LoRA

Are the fine-tuning updates full rank ?



LoRA

What is full rank vs low rank matrix ?



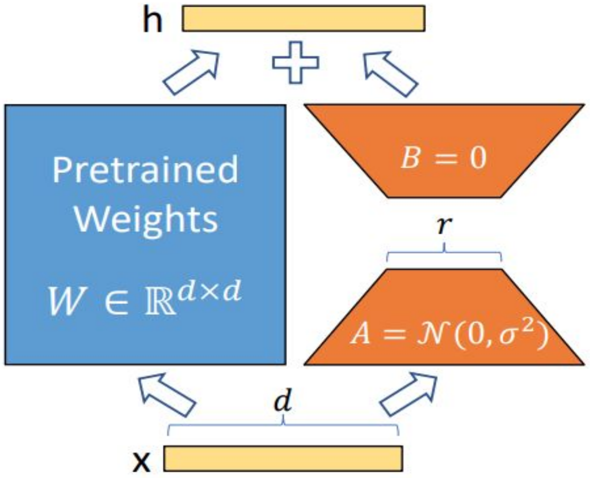
- Rank of a matrix, (r) is the number of linearly independent columns/rows
- For a full rank matrix, r = lowest dimension of the matrix

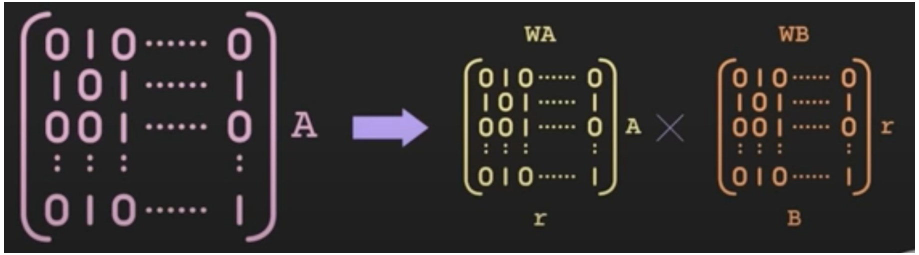
$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

https://bobbielf2.github.io/blog/2021/03/21/low-rank-and-rank-structured-matrices/
https://www.3blue1brown.com/lessons/inverse-matrices

LoRA

Use low rank decomposition for fine tuning updates



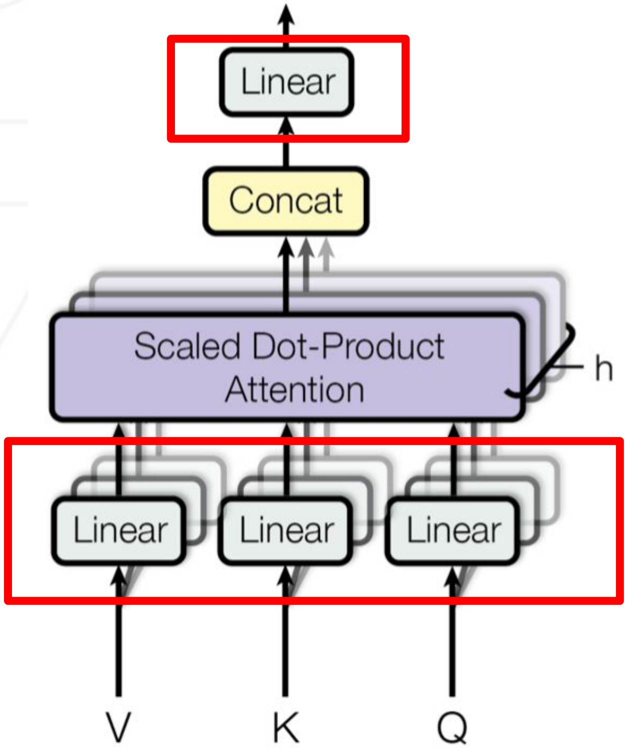


$$h = W_0x + \Delta W x = W_0 x + BAx$$

$$B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, r \ll \min(d,k)$$

Georgia Tech

LoRA for Transformer



- Weight matrices of linear layers in Transformer achtitecture:
  - W_Q, W_V, W_k, W_O

- Apply Low rank decomposition to these matrices while fine tuning
  - $W_Q = [\,]_{100,100} = B_{100,3}\ A_{3,100}$

$3*100*2$ vs 100^2

Experimental Analysis

Experimental Setup

Baseline methods

- Fine-tune
- Bias only
- Prefix-embedding tuning
  - injects special tokens alongside the input tokens
- Prefix-layer tuning
  - learn the Prefix-embedding after every layer.
- Adapter tuning
  - inserts adapter layers between the self-attention module (and the MLP module) and the subsequent residual connection.

Baseline models

- BERT
- **RoBERTa**
- GPT 2
- **GPT 3**

Results: RoBERTa

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	87.1$_{\pm.0}$	94.2$_{\pm.1}$	88.5$_{\pm1.1}$	60.8$_{\pm.4}$	93.1$_{\pm.1}$	90.2$_{\pm.0}$	71.5$_{\pm2.7}$	89.7$_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	87.3$_{\pm.1}$	94.7$_{\pm.3}$	88.4$_{\pm.1}$	62.6$_{\pm.9}$	93.0$_{\pm.2}$	90.6$_{\pm.0}$	75.9$_{\pm2.2}$	90.3$_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	87.5$_{\pm.3}$	**95.1**$_{\pm.2}$	89.7$_{\pm.7}$	63.4$_{\pm1.2}$	**93.3**$_{\pm.3}$	90.8$_{\pm.1}$	**86.6**$_{\pm.7}$	**91.5**$_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.9**$_{\pm1.2}$	**68.2**$_{\pm1.9}$	**94.9**$_{\pm.3}$	91.6$_{\pm.1}$	**87.4**$_{\pm2.5}$	**92.6**$_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	90.2$_{\pm.3}$	96.1$_{\pm.3}$	90.2$_{\pm.7}$	**68.3**$_{\pm1.0}$	**94.8**$_{\pm.2}$	**91.9**$_{\pm.1}$	83.8$_{\pm2.9}$	92.1$_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	**90.5**$_{\pm.3}$	**96.6**$_{\pm.2}$	89.7$_{\pm1.2}$	67.8$_{\pm2.5}$	**94.8**$_{\pm.3}$	91.7$_{\pm.2}$	80.1$_{\pm2.9}$	91.9$_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	89.9$_{\pm.5}$	96.2$_{\pm.3}$	88.7$_{\pm2.9}$	66.5$_{\pm4.4}$	94.7$_{\pm.2}$	92.1$_{\pm.1}$	83.4$_{\pm1.1}$	91.0$_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	90.3$_{\pm.3}$	96.3$_{\pm.5}$	87.7$_{\pm1.7}$	66.3$_{\pm2.0}$	94.7$_{\pm.2}$	91.5$_{\pm.1}$	72.9$_{\pm2.9}$	91.5$_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.2**$_{\pm1.0}$	68.2$_{\pm1.9}$	**94.8**$_{\pm.3}$	91.6$_{\pm.2}$	**85.2**$_{\pm1.1}$	**92.3**$_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	**91.9**$_{\pm.2}$	96.9$_{\pm.2}$	**92.6**$_{\pm.6}$	**72.4**$_{\pm1.1}$	**96.0**$_{\pm.1}$	**92.9**$_{\pm.1}$	**94.9**$_{\pm.4}$	**93.0**$_{\pm.2}$	**91.3**

Table 2: $\text{RoBERTa}_{\text{base}}$, $\text{RoBERTa}_{\text{large}}$, and $\text{DeBERTa}_{\text{XXL}}$ with different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics. * indicates numbers published in prior works. † indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

Results: GPT-3

Method	# of Trainable Parameters	WikiSQL Accuracy (%)	MNLI-m Accuracy (%)	SAMSum R1/R2/RL
GPT-3 175B (Fine-Tune)	175,255.8M	73.0	89.5	52.0/28.0/44.5
GPT-3 175B (Bias Only)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 175B (PrefixEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 175B (PrefixLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 175B (LoRA)	4.7M	73.4	91.3	52.1/28.3/44.3
GPT-3 175B (LoRA)	37.7M	**73.8**	**91.7**	**53.2/29.2/45.0**

Table 1: Logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched and Rouge-1/2/L on SAMSum achieved by different GPT-3 adaptation methods. LoRA performs better than prior approaches, including conventional fine-tuning. The result on WikiSQL has a fluctuation of $\pm 0.3\%$ and MNLI-m $\pm 0.1\%$.

Ablation Study: Understanding Low-rank Update

Q1: Which weight to apply LoRA?

	# of Trainable Parameters = 18M						
Weight Type Rank r	W_q 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

- Training **at least W_q and W_v** to achieve good results
- Training all attention weights gives the best results on the same parameter budget

*Q2: How to choose rank **r** for LoRA?*

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

- Only **r = 1** already enable very good results

****Results with GPT-3**

Strengths and Weaknesses

Strengths and Weaknesses

Strengths

1. Innovative approach to fine-tuning with multiple practical advantages

   - Save on storage of multiple fine-tune models
   - Can merge the weight during inference, thus requiring no additional latency
   - Lower hardware requirements for tuning

2. Ablation studies prove that low-rank adaptation is effective, even with r = 1

3. A general method, can be applied to many problems and in combinations with other fine-tuning methods

Weaknesses

1. Lower hardware requirements, **but still high**: At least GPUs that have enough RAM to load the full base models

2. Deployment still needs same amount of memory (say 175B), only if you are deploying multiple models, do the savings kick in.

3. Adds decomposition rank ‘r’ as another hyperparameter to tune.

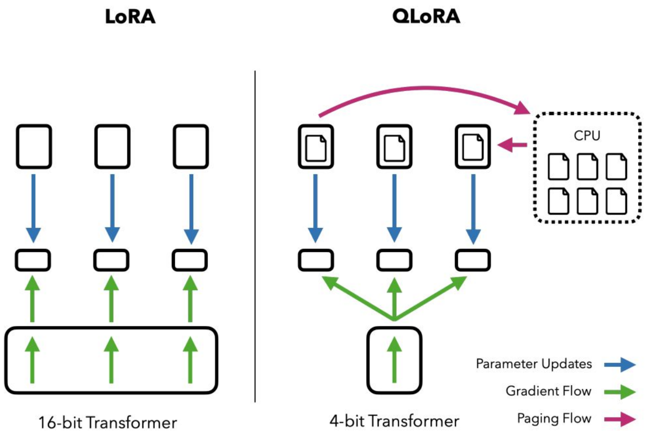
4. Ablation missing for cases where different weight matrices have different rank decompositions, say 2 or Q, 6 for V.

Further developments

Recent developments leveraging LoRA

Quantization (QLoRA)

Dettmers et al. (2023) combine LoRA with 4-bit quantization to *reduce the memory requirement* and improve *computation efficientcy* of fine tuning without sacrificing performance



Fine-tune Diffusion model*

- 2x faster fine-tuning of the Stable Diffusion model compared to Dreambooth
- Small model (1MB ~ 6MB vs GBs), enable sharing

Photo-realistic images

Cartoon images

* cloneofsimo, Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning, https://github.com/cloneofsimo/lora 2023

Georgia Tech

Q&A