LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models

Yixiao Li[1*], Yifan Yu[1*], Chen Liang[*], Pengcheng He[◇],
Nikos Karampatziakis[◇], Weizhu Chen[◇], Tuo Zhao[*]

[*]Georgia Institute of Technology, [◇]Microsoft Azure AI, [1]Equal contribution

Presenter: Xiaodong Liu

May 08, 2024

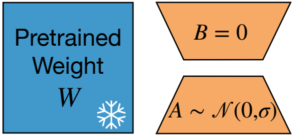
Finetune Generative AI on Your PC

Stable Diffusion MISTRAL AI_ ϕ Phi LLaMA

- Focus on specific domain.
- Protect your personal and private data.
- Transfer to your own writing or painting style.
- ...

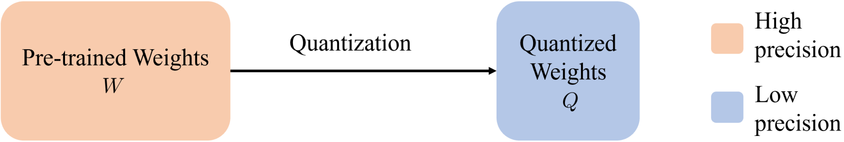
Challenges of Full Model Finetuning

Impossible to finetune a 7B model on one RTX 4080 (16BG).



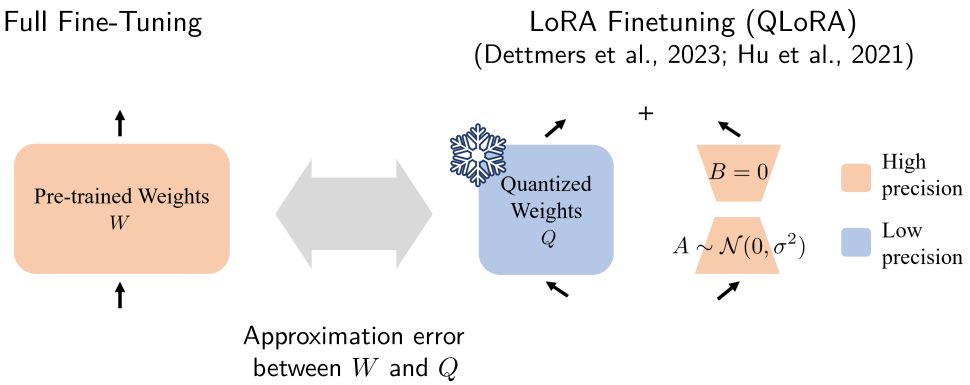
		LoRA (Hu et al., 2021)
Memory of the model	7B x 16bit = 14GB	7B x 16bit = 14GB
Memory of grad and optim states	7B x 32bit x 3 = 84GB	5% x 7B x 32bit x 3 = 4.2GB
GPU Footprint	108GB	>27GB

Quantization: Low-Precision Storage



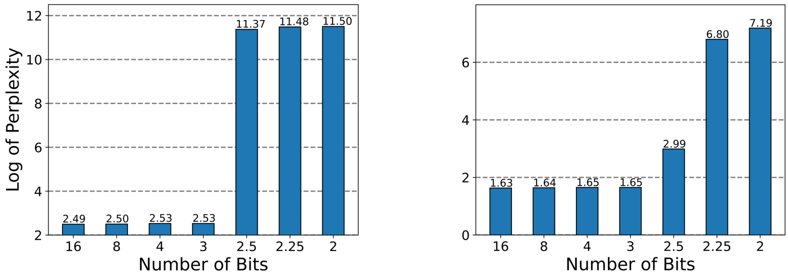
	FP32	FP16	4 bit	2 bit
70B Inference	280GB	140GB	$\sim$35GB	$\sim$17GB
	(4x80G GPU)	(2x80G)	(1x40G)	(1x24G)

Discrepancy betw. Pre-trained Weights and LoRA Initialization



Discrepancy betw. Pre-trained Weights and LoRA Initialization

Evaluation perplexity (the lower the better) of applying LoRA to a quantized LLaMA-2-13b under different bit-levels on WikiText-2 dataset.



At LoRA Initialization

After LoRA Finetuning

LoftQ: LoRA-Fine-Tuning-Aware Quantization

Find quantized weights and low-rank weights such that the low-rank weights can bridge the discrepancy between the quantized weights and the pre-trained weights:

$$\min_{Q,A,B} \|W - Q - AB^\top\|_F^2,$$

where W: pre-trained weights; Q: quantized weights; A, B: low-rank approximation; $\|\cdot\|_F$: Frobenius norm.

Algorithm: Alternating Optimization

Input: T: # of iterations; $q(\cdot)$: quantization function.
$A_0 \leftarrow 0, B_0 \leftarrow 0$.
for t $= 1$ to T **do**
  Quantization: $Q_t \leftarrow q(W - A_{t-1}B_{t-1}^{\top})$.
  Low-rank approximation: $A_t, B_t \leftarrow \texttt{Truncated-SVD}(W - Q_t)$.
end for
Output: Q_T, A_T, B_T.

- Use Q_T, A_T, B_T as the initialization for LoRA fine-tuning.
- No limit of quantization methods.
- Alternating optimization reinforces Q and A, B.
- Without calibration data.
- Apply it to different weights in parallel.

Encoder-Only: DeBERTaV3-base on NLU

Method	Quantization	LoRA Rank	MNLI m / mm	QNLI Acc	RTE Acc	SST-2 Acc	SQuAD v2 F1	ANLI Acc
Fine-Tune	-	-	90.5/90.6	94.0	82.0	95.3	92.8	59.8
LoRA	-	16	90.4/90.5	94.6	85.1	95.1	93.1	60.2
QLoRA	2-bit NormalFloat	16	75.4/75.6	82.4	55.9	86.5	71.2	N/A
LoftQ (our)	2-bit NormalFloat	16	**84.7/85.1**	**86.6**	**61.4**	**90.2**	**88.6**	**47.1**
QLoRA	2-bit Uniform	16	76.5/76.3	83.8	56.7	86.6	77.6	-
LoftQ (our)	2-bit Uniform	16	**87.3/87.1**	**90.6**	**61.1**	**94.0**	**91.2**	-

"N/A": model does not converge.

Encoder-Decoder: BART-large on Summarization

Method	Quantization	LoRA Rank	XSum ROUGE-1/2/L	CNN/DailyMail ROUGE-1/2/L
Lead 3	-	-	16.30/1.60/11.95	40.42/17.62/36.67
Fine-Tune			45.14/22.27/37.25	44.16/21.28/40.90
LoRA	-	16	43.95/20.72/35.68	45.03/21.84/42.15
QLoRA	4-bit NormalFloat	16	43.29/20.05/35.15	43.42/20.62/40.44
LoftQ (our)			**44.51/21.14/36.18**	**43.96/21.06/40.96**
QLoRA	4-bit Uniform	16	42.45/19.36/34.38	43.00/20.19/40.02
LoftQ (our)			**44.29/20.90/36.00**	**43.87/20.99/40.92**

Decoder-Only: LLaMA-2 on NLG

			LLaMA-2-7b		LLaMA-2-13b	
Method	**Quantization**	**LoRA Rank**	**WikiText-2** PPL↓	**GSM8K** Acc↑	**WikiText-2** PPL↓	**GSM8K** Acc↑
LoRA	-	64	5.08	34.4	5.12	45.3
QLoRA	4-bit NormalFloat	64	5.70	**35.1**	5.22	39.9
LoftQ (our)	4-bit NormalFloat	64	**5.24**	35.0	**5.16**	**45.0**
QLoRA	3-bit NormalFloat	64	5.73	32.1	5.22	40.7
LoftQ (our)	3-bit NormalFloat	64	**5.63**	**32.9**	**5.13**	**44.4**
QLoRA	2-bit NormalFloat	64	N/A	N/A	N/A	N/A
LoftQ (our)	2-bit NormalFloat	64	**7.85**	**20.9**	**7.69**	**25.4**

N/A: model does not converge. Regularization applied. 3 bit is mixed precision.

Phi-2 and LLAMA-3 on GSM8K

Model	Bits	Rank	Method	GSM8K
Phi-2 (2.7B)	16	-	Full FT	66.8 ± 1.2
Phi-2 (2.7B)	16	64	LoRA	64.8 ± 0.5
Phi-2 (2.7B)	4	64	QLoRA	60.2 ± 0.6
Phi-2 (2.7B)	4	64	LoftQ	$\mathbf{64.1} \pm \mathbf{0.7}$
LLAMA-3-8B	16	-	Full FT	70.4 ± 0.7
LLAMA-3-8B	16	64	LoRA	69.3 ± 1.5
LLAMA-3-8B	4	64	QLoRA	67.4 ± 1.0
LLAMA-3-8B	4	64	LoftQ	$\mathbf{68.0} \pm \mathbf{0.6}$

Memory Footprint

Model	Parameters	GPU Memory
Phi-2	$\sim$3B	$<$11GB (e.g., RTX 2080 Ti, RTX 3080)
Mistral-7B	$\sim$7B	$<$16GB (e.g., RTX 4080)

LoRA fine-tuning on GSM8K, input length $=$ 1024, batch size $=$ 1

Use LoftQ's Weight Init with Simple Code Changes

```
...
backbone_model = AutoModelForCausalLM.from_pretrained(
    args.model_name_or_path, # e.g., "LoftQ/Llama-2-7b-hf-4bit-64rank" on HF
    quantization_config=BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
        ...
    ),
)
model = PeftModel.from_pretrained(
  backbone_model,
  args.model_name_or_path,
  subfolder="loftq_init", # subfolder storing weights for A, B
  is_trainable=True)
...
model.train()
```

Initialize the quantized backbone using LoftQ

Initialize LoRA adaptors using LoftQ

Off-the-Shelf Models

- Llama-2 (7B, 13B, 70B), CodeLlama(7B, 13B, 70B)
- Phi-2
- Mistral-7B
- Llama-3 (8B, 70B), Llama-3-Instruct (8B, 70B)
- Phi-3 (mini, ...)

Future Directions

- Recap the objective:

$$\min_{Q,A,B} \|W - Q - AB^\top\|_{\mathrm{F}}^2.$$

- Can we do better than alternating optimization? For example, differential quantization.
- Can we do better with calibration data? For example,

$$\min_{Q,A,B} \|X(W - Q - AB^\top)\|_{\mathrm{F}}^2.$$

- Can we do better with adaptive low-rank approximation? For example, AdaLoRA (Zhang et al., 2022).

Take Home Messages

- Naive quantization hurts LoRA performance.
- LoftQ mutually reinforces the quantized backbone Q and init LoRA adapter A, B.
- LoftQ is an easy replacement of QLoRA.
- We have more off-the-shelf quantized models on `huggingface/LoftQ`.

Thank you!

Scan me to read more

We are releasing more off-the-shelf models on HuggingFace.