# QLoRA: Efficient Finetuning of Quantized LLMs

**最先端NLP勉強会**

2023年8月27日

京都大学大学院 情報学研究科 知能情報学コース 言語メディア研究室 (黒橋研)
D1　清水周一郎

# 自己紹介

- 京大 電気電子工学科 (学士) → 情報学研究科 (修士・博士)
- 趣味: 将棋、ピアノ、フィギュアスケート、ボルダリング、卓球など
- JSPS DC1 (2023.4 - 2026.3)
  - **メタ介入機構を備えた音声対話翻訳システムの構築**
- Major Publications
  - <u>S. Shimizu</u>, C. Chu, S. Li, S. Kurohashi. **Towards Speech Dialogue Translation Mediating Speakers of Different Languages.** Findings of ACL 2023.
  - <u>S. Shimizu</u>, C. Chu, S. Li, S. Kurohashi. **Cross-Lingual Transfer Learning for End-to-End Speech Translation.** 自然言語処理 Vol. 29 No. 2, 2022.
  - Y. Li, <u>S. Shimizu</u>, W. Gu, C. Chu, S. Kurohashi. **VISA: An Ambiguous Subtitles Dataset for Visual Scene-Aware Machine Translation.** LREC 2022.

<a href="https://cromz22.github.io/">https://cromz22.github.io/</a>

# Finetuning is Expensive!

| Model | Finetuning Memory |
|---:|---:|
| T5-11B | 176 GB |
| LLaMA2-33B | 396 GB |
| LLaMA2-70B | 140 GB |

- Nvidia GPUs
  - A100: 40/80 GB
  - A6000: 49 GB
  - V100: 32 GB

# Finetuning is Expensive!

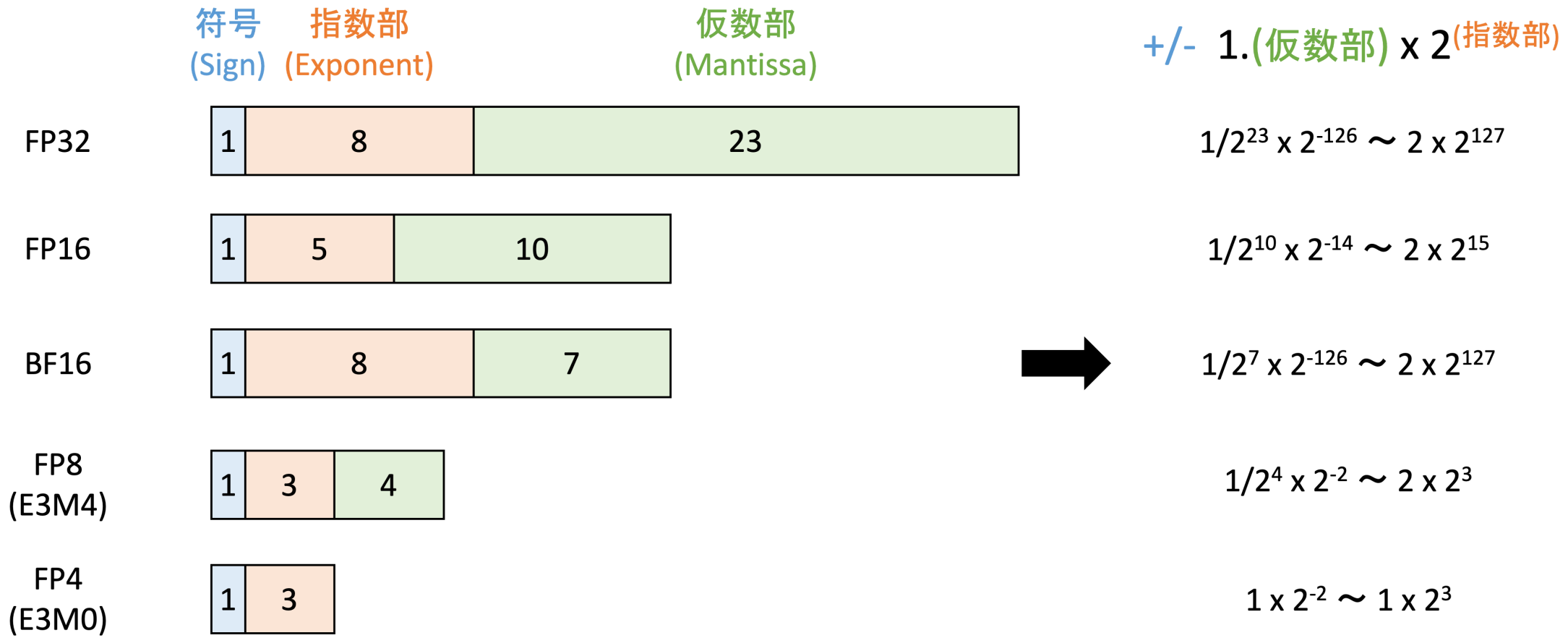
| Model | Finetuning Memory | Finetuning Memory (QLoRA) |
|---|---|---|
| T5-11B | 176 GB | 6 GB |
| LLaMA2-33B | 396 GB | 23 GB |
| LLaMA2-70B | 140 GB | 46 GB |

- Nvidia GPUs
  - A100: 40/80 GB
  - A6000: 49 GB
  - V100: 32 GB

# QLoRA: Efficient Finetuning of Quantized LLMs

- Techniques to save memory:
    i. 4-bit NormalFloat (NF4) data type
    ii. Double Quantization
    iii. Paged Optimizers
- Findings through creating the **Guanaco** model family:
    - Dataset suitability matters more than size for a given task
    - GPT-4 evaluation can work but also has some issues

# Preliminary: Floating Point Representation



# Data Type as a Map

- All data type can be represented in the form of a map
  `{index: "normalized" value}`
  - e.g., int4

| index | 0 | 1 | 2 | ... | 14 |
|---|---|---|---|---|---|
| value | -7 | -6 | -5 | ... | 7 |
| "normalized" value | -1 | -0.86 | -0.71 | ... | 1 |

- Defining a data type is equivalent to defining this map

# Example of Quantization: Strange 2-bit Type

---

- To **quantize** $[10, -3, 5, 6]$:
  i. Normalize with absmax $\rightarrow [1, -0.3, 0.5, 0.6]$
  ii. Find the closest value in the map $\rightarrow [1.0, \mathbf{0.3}, 0.5, \mathbf{0.5}]$
  iii. Use the mapping $\rightarrow [3, 1, 2, 2]$
- To **dequantize** $[3, 1, 2, 2]$:
  i. Use the mapping $\rightarrow [1.0, 0.3, 0.5, 0.5]$
  ii. Denormalize with memorized absmax $\rightarrow [10, \mathbf{3}, 5, \mathbf{5}]$

# Quantization as a Mapping

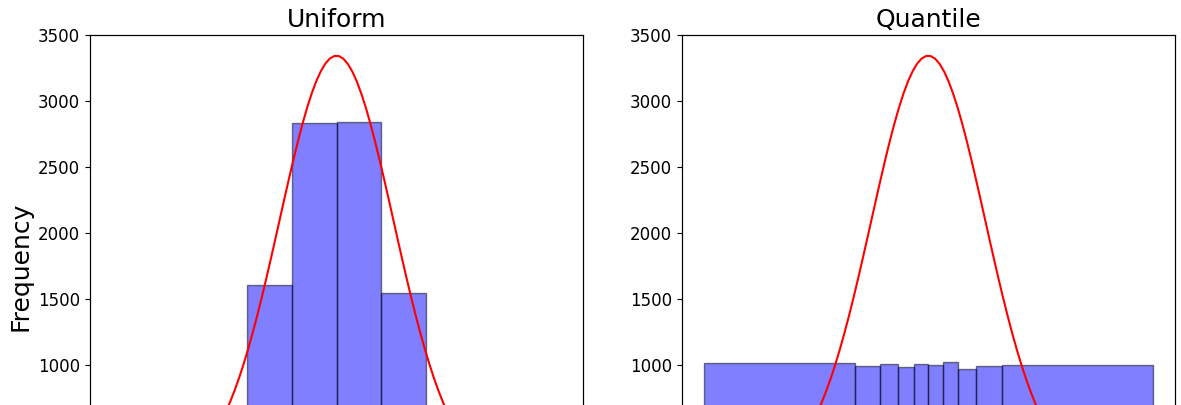
Given a tensor $X \in \mathbb{R}^{b \times h}$ of any data type, to quantize them in to another data type:

1. "Normalize" $X$ into the range $[-1.0, 1.0]$
2. Find the closest value in the map of the target data type

- Naive quantization
  - Divide $X$ by abs$(X)$
- Block-wise quantization (Dettmers et al., ICLR 2022) for outliers
  i. Chunk $X$ into $n$ blocks
  ii. Quantize each block independently wiht $n$ quantization constants

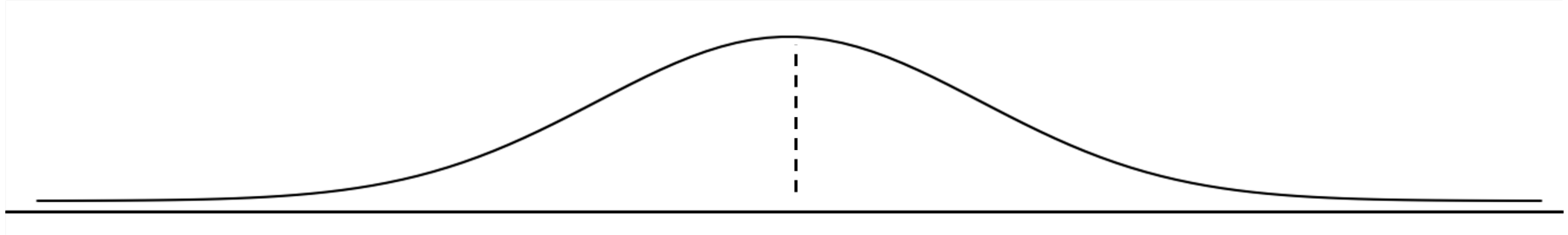
# Quantile quantization

- $q$-quantile: divides a distribution by $q : 1 - q$
- $k$-bit quantile quantization: information theoretically optimal (Dettmers et al., 2022)
    i. Find $2^k + 1$ quantiles
    ii. Use the $2^k$ median values as bin centers



# 4-bit NormalFloat (NF4) Data Type

1. Consider a normal distribution with mean 0, std 1



2. Get 7 quantiles on the left, 8 on the right

   [-1.85, -1.29, -0.07, -0.73, -0.53, -0.34, -0.17]  [0.15, 0.30, 0.45, 0.62, 0.81, 1.04, 1.34, 1.85]

3. Concatenate with a zero in between

   [-1.85, -1.29, -0.07, -0.73, -0.53, -0.34, -0.17, 0, 0.15, 0.30, 0.45, 0.62, 0.81, 1.04, 1.34, 1.85]

4. Normalize (divide by absmax)

   [ -1.0,  -0.7, -0.53, -0.39,  -0.28, -0.18, -0.09, 0, 0.08, 0.16, 0.25, 0.34, 0.44, 0.56, 0.72,  1.0]

# Other Memory-saving Techniques

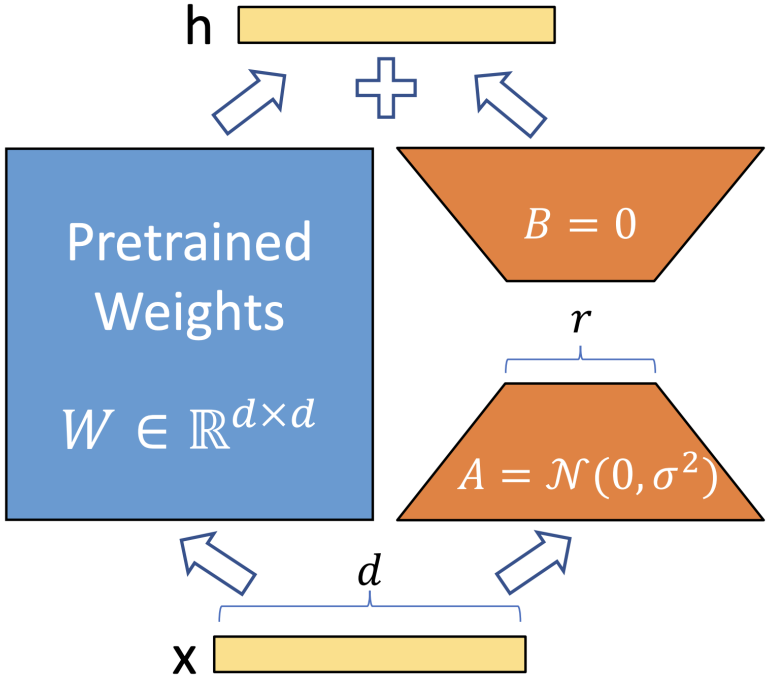
- **Double quantization**
  - First quantization constants as inputs to a second quantization
  - Before: 32-bit constants with blocksize 64
    $\rightarrow 32/64 = 0.5$ bits/param
  - After: 8-bit quantization with blocksize 256
    $\rightarrow 8/64 + 32/(64 \cdot 256) = 0.127$ bits/param ($0.373$ bits/param reduction)
- **Paged optimizer**
  - Fall-back to CPU when GPU runs out of memory

# Preliminary: LoRA

LoRA: Low-Rank Adaptation of Large Language Models (Hu et al., ICLR 2022)

- Freeze the pretrained model weights and inject trainable rank decomposition matrices
- Greatly reduce the number of trainable parameters

$$
\begin{aligned}
Y &= X(W + sAB) \\
&= XW + sXAB \\
&(s \in \mathbb{R}, A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times d})
\end{aligned}
$$



# QLoRA

$$Y^{\text{BF16}} = X^{\text{BF16}} W^{\text{BF16}} + sX^{\text{BF16}} A^{\text{BF16}} B^{\text{BF16}}$$

Calculations are done in BF16

(because $\frac{\partial X}{\partial W}$ needs to be calculated in BF16)

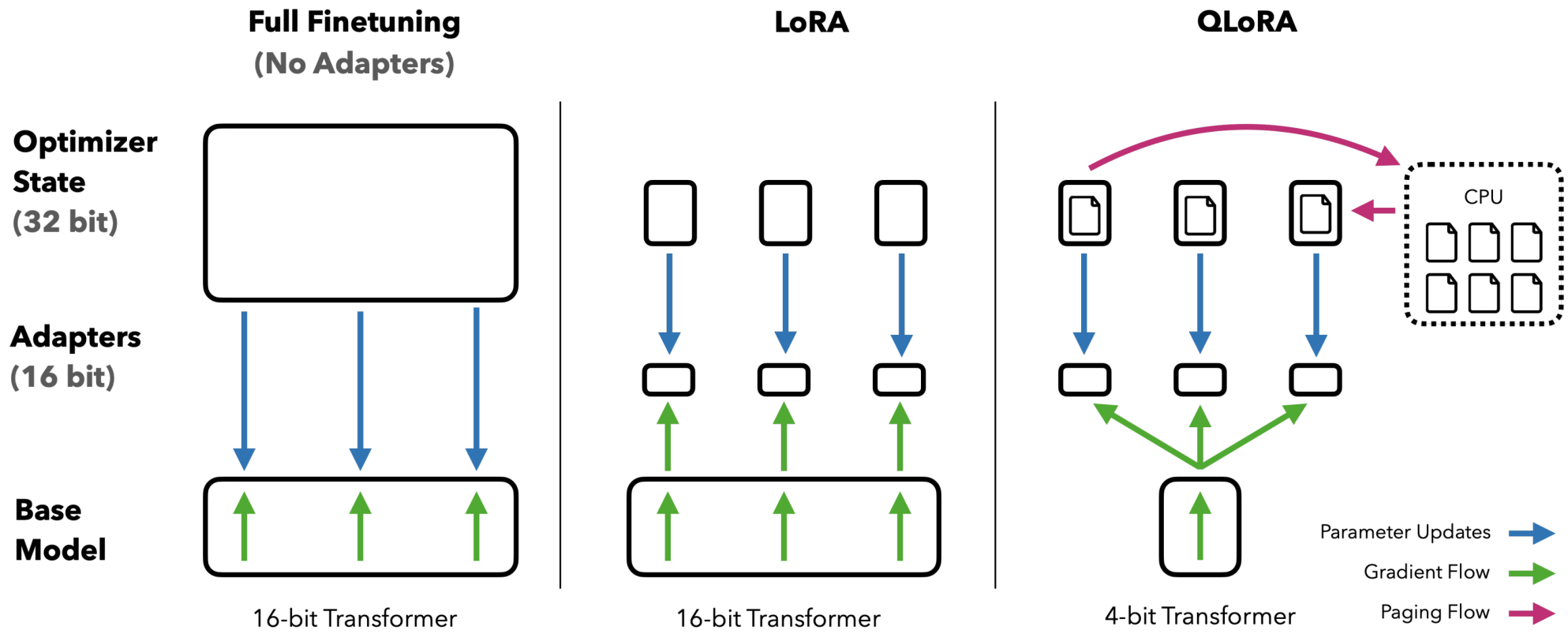
$$W^{\text{BF16}} = \text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, W^{\text{NF4}})$$

by dequantizing the quantized pretrained weights

$$= \text{dequant}(\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}), W^{\text{NF4}})$$

by dequantizing the constants for the quantization of W

# QLoRA: Overview



# Can QLoRA perform as well as full-model fintuning?

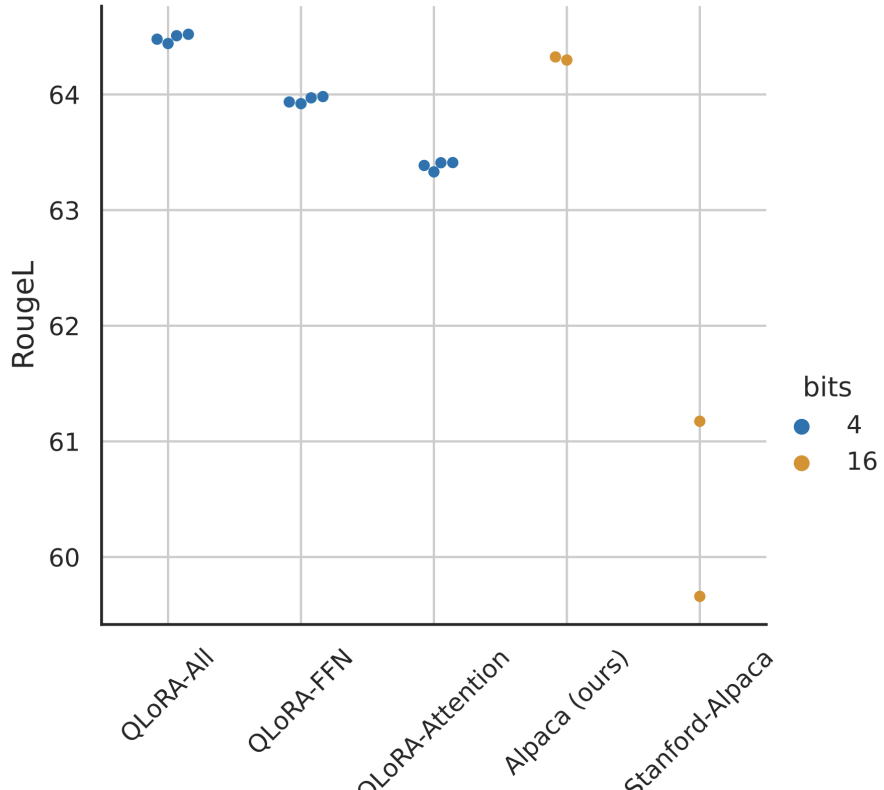
**Table 3:** Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

| Dataset | GLUE (Acc.) | Super-NaturalInstructions (RougeL) | | | | |
|---|---|---|---|---|---|---|
| Model | RoBERTa-large | T5-80M | T5-250M | T5-780M | T5-3B | T5-11B |
| BF16 | 88.6 | 40.1 | 42.1 | 48.0 | 54.3 | 62.0 |
| BF16 replication | 88.6 | 40.0 | 42.2 | 47.3 | 54.9 | - |
| LoRA BF16 | 88.8 | 40.5 | 42.6 | 47.1 | 55.4 | 60.7 |
| QLoRA Int8 | 88.8 | 40.4 | 42.9 | 45.4 | 56.5 | 60.7 |
| QLoRA FP4 | 88.6 | 40.3 | 42.4 | 47.5 | 55.6 | 60.9 |
| QLoRA NF4 + DQ | - | 40.4 | 42.7 | 47.7 | 55.3 | 60.9 |

- Performance lost due to imprecise quantization can be fully recoverd through finetuning

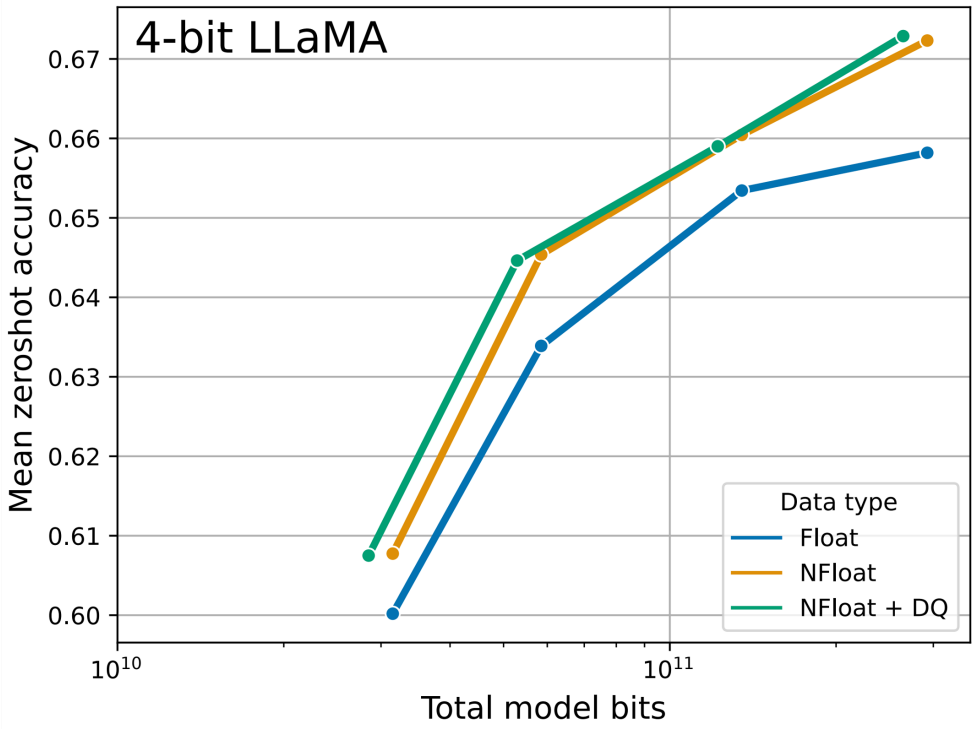
# On what parameters should we apply LoRA?

- Finetune LLaMA 7B on Alpaca dataset
- LoRA on all layers are required to match full finetuning performance



# Is NF4 better than Float4?

Average acc. of Winogrande, HellaSwag, PiQA, Arc-Easy, and ArcChallenge




- NF4 is better than FP4

# Chatbot Experiments

- Datasets
  - **Crowd-sourced**: OASST1, HH-RLHF
  - **Distillation**: Alpaca, self-instruct, unnatural instructions
  - **Corpora aggregations**: Flan v2
  - **Hybrids**: Chip2, Longform
- Setup
  - Finetuning with cross-entropy loss (w/o RLHF)
- Baselines
  - Research: Vicuna, Open Assistant
  - Commercial: GPT-4, GPT-3.5-turbo, Bard (Note: as of May 2023)

# Chatbot Experiments: Evaluation

- Benchmarks
  - **MMLU**: multiple-choice benchmark covering 57 tasks
  - **Vicuna** prompts: a set of 80 prompts from a diverse set of categories
  - **OASST1** validation set: multilingual collection of crowd-sourced multi-turn dialogues between a user and an assistant, including 953 user queries
- Automated evaluation
  - GPT-4 evaluates a system's response and ChatGPT's response with scores 1-10
- Human evaluation

# Different Datasets for Different Tasks

- FLAN v2 performs the best for MMLU, but Guanaco (OASST1) is better for chat
  → Different datasets should be used for different tasks

**Table 5:** MMLU 5-shot test results for different sizes of LLaMA finetuned on the corresponding datasets using QLoRA.

| Dataset | 7B | 13B | 33B | 65B |
|---|---|---|---|---|
| LLaMA no tuning | 35.1 | 46.9 | 57.8 | 63.4 |
| Self-Instruct | 36.4 | 33.3 | 53.0 | 56.7 |
| Longform | 32.1 | 43.2 | 56.6 | 59.7 |
| Chip2 | 34.5 | 41.6 | 53.6 | 59.8 |
| HH-RLHF | 34.9 | 44.6 | 55.8 | 60.1 |
| Unnatural Instruct | 41.9 | 48.1 | 57.3 | 61.3 |
| Guanaco (OASST1) | 36.6 | 46.4 | 57.0 | 62.2 |
| Alpaca | 38.8 | 47.8 | 57.3 | 62.5 |
| FLAN v2 | 44.5 | 51.4 | 59.2 | 63.9 |

**Table 6:** Zero-shot Vicuna benchmark scores as a percentage of the score obtained by ChatGPT evaluated by GPT-4. We see that OASST1 models perform close to ChatGPT despite being trained on a very small dataset and having a fraction of the memory requirement of baseline models.

| Model / Dataset | Params | Model bits | Memory | ChatGPT vs Sys | Sys vs ChatGPT | Mean | 95% CI |
|---|---|---|---|---|---|---|---|
| GPT-4 | - | - | - | 119.4% | 110.1% | **114.5%** | 2.6% |
| Bard | - | - | - | 93.2% | 96.4% | 94.8% | 4.1% |
| **Guanaco** | 65B | 4-bit | 41 GB | 96.7% | 101.9% | **99.3%** | 4.4% |
| Alpaca | 65B | 4-bit | 41 GB | 63.0% | 77.9% | 70.7% | 4.3% |
| FLAN v2 | 65B | 4-bit | 41 GB | 37.0% | 59.6% | 48.4% | 4.6% |

# Elo Rankings

**Table 7:** Elo rating for a tournament between models where models compete to generate the best response for a prompt, judged by human raters or GPT-4. Overall, Guanaco 65B and 33B tend to be preferred to ChatGPT-3.5 on the benchmarks studied. According to human raters they have a Each 10-point difference in Elo is approximately a difference of 1.5% in win-rate.

| Benchmark | Vicuna | | Vicuna | | Open Assistant | | |
|---|---|---|---|---|---|---|---|
| # Prompts | 80 | | 80 | | 953 | | |
| Judge | Human raters | | GPT-4 | | GPT-4 | | Median Rank |
| Model | Elo | Rank | Elo | Rank | Elo | Rank | |
| GPT-4 | 1176 | 1 | 1348 | 1 | 1294 | 1 | 1 |
| Guanaco-65B | 1023 | 2 | 1022 | 2 | 1008 | 3 | 2 |
| Guanaco-33B | 1009 | 4 | 992 | 3 | 1002 | 4 | 4 |
| ChatGPT-3.5 Turbo | 916 | 7 | 966 | 5 | 1015 | 2 | 5 |
| Vicuna-13B | 984 | 5 | 974 | 4 | 936 | 5 | 5 |
| Guanaco-13B | 975 | 6 | 913 | 6 | 885 | 6 | 6 |
| Guanaco-7B | 1010 | 3 | 879 | 8 | 860 | 7 | 7 |
| Bard | 909 | 8 | 902 | 7 | - | - | 8 |

- Moderate agreement between GPT-4 and human annotators (Kendall $\tau = 0.43$, Spearman $r = 0.55$)

# Conclusion

- Present QLoRA that can perform as good as full-finetuning, which saves memory with:
    i. 4-bit NormalFloat (NF4) data type
    ii. Double Quantization
    iii. Paged Optimizers
- Findings through creating the **Guanaco** model family:
    - Dataset suitability matters more than size for a given task
    - GPT-4 evaluation can work but also has some issues

# References


- LoRA: Low-Rank Adaptation of Large Language Models (Hu et al., ICLR 2022)
- 8-bit Optimizers via Block-wise Quantization (Dettmers et al., ICLR 2022)
- LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (Dettmers et al., NeurIPS 2022)
- The case for 4-bit precision: k-bit Inference Scaling Laws (Dettmers and Zettlemoyer, ICML 2023)
- Youtube videos (video1: slide1, video2: slide2) by the author
- A Gentle Introduction to 8-bit Matrix Multiplication for transformers at scale using Hugging Face Transformers, Accelerate and bitsandbytes
- Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA
- イロレーティング - Wikipedia

# Appendix: Elo Rating

あるプレイヤーが平均的プレイヤーと対戦した場合の勝利, 敗北確率をそれぞれ $W$, $L$ とする

$$R = 400 \log_{10} \frac{W}{L} + R_0$$

勝敗比が積により推移するという仮定をおくと、$W_{AB} = \frac{1}{10^{(R_B - R_A)/400} + 1}$

---

(例えば、藤井聡太七冠 2079、永瀬拓也王座 1890のとき藤井七冠が勝利する確率は 74.8%)

更新式: $R_A \leftarrow R_A + K\left(\mathrm{Wins} - \mathrm{Games} \times W_{AB}\right)$