Compressing Large Language Models using Low Rank and Low Precision Decomposition

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J. Goldsmith, Mert Pilanci

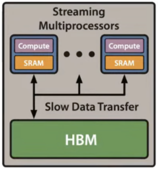
Stanford ENGINEERING Electrical Engineering

PRINCETON Electrical and Computer Engineering

Dec 9 - Dec 15, 2024
Vancouver

Compressing Large Language Models

[Credits: GPT-4 + DALL.E 3]



[Credits: FlashAttention, Dao et. al.]

- LLMs are memory hungry and often cannot be loaded on consumer GPUs: Eg: LLaMa 70B in BF16 takes up 140 GiB. Consumer GPUs (eg. NVIDIA A10G) have only 24 GiB of HBM.
- High inference latency (fewer tokens per second): Inference with low batch sizes is typically memory bound, i.e., back-and-forth communication between GPU HBM and SRAM is the bottleneck.
- Out-of-memory (OOM) issues while finetuning: Fine-tuning LLMs requires storing weights, activations, and optimizer states.
- Communication bandwidth becomes a bottleneck in distributed inference using multi-GPU (eg. NVLink) or multi-node (eg. InfiniBand).
- Increased model sharing latency (HuggingFace upload/download)
- **Goal of our work: Compress an LLM while preserving its accuracy.**

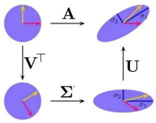
Low Rankness of LLM weights

- LLM weights (Query, Key, ...) are represented as matrices. Matrices are linear transforms on input activations. While compressing a weight matrix, we should preserve this functionality.

- **Singular value decomposition**: Any matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ can written as:

$$\mathbf{A} = \sum_{i=1}^{\mathrm{rank}(\mathbf{A})} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top,$$

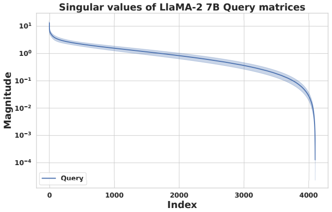
where $\{\sigma_i\}$ are the singular values, and $\mathbf{u}_i \in \mathbb{R}^n$, $\mathbf{v}_i \in \mathbb{R}^d$ are singular vectors.



$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

- Higher singular value components majorly capture how input activations are transformed into output activations for each layer in a forward pass.

We leverage the top singular components to compress weight matrices by obtaining an approximate low-rank structure!



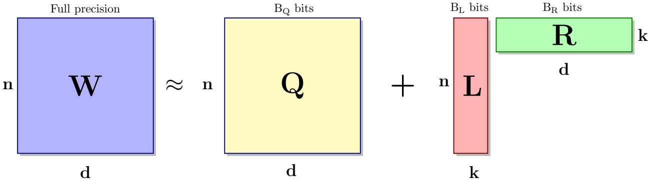
Low-precision representations

[Credits: GPT-4 + DALL.E 3]

- Low-precision formats also reduce memory footprint by using fewer bits to represent real numbers.
  Eg. INT4, FP4, MXFP4, ...
- Low-precision compute is faster.
  Eg. NVIDIA H100 specs: 1979 teraFLOPS with BFLOAT16 vs. 3958 teraFLOPS with FP8.
- Low-precision operations also require fewer Watts, i.e., more energy efficient.

Low-precision and Low-Rank Decomposition

Problem: **How to jointly obtain a low rank as well as low precision approximation of a matrix?**



Calibration Aware Low-Precision and Low-Rank Decomposition

$$\min_{\mathbf{Q},\mathbf{L},\mathbf{R}} \|(\mathbf{Q}+\mathbf{L}\mathbf{R}-\mathbf{W})\mathbf{X}^{\top}\|_{\mathrm{F}}^2$$ subject to $\mathbf{Q}, \mathbf{L}, \mathbf{R}$ using $\mathrm{B_Q}$, $\mathrm{B_L}$, and $\mathrm{B_R}$ bits respectively.

- Calibration data $\mathbf{X}$: Sampled from RedPajama dataset.
- Low-rank factors $\mathbf{L}$ and $\mathbf{R}$ capture the large singular components of $\mathbf{W}$ with fewer parameters but high fidelity ($\mathrm{B_L} = \mathrm{B_R} = 4$ bits).
- Full-rank backbone $\mathbf{Q}$ is quantized aggressively ($\mathrm{B_Q} = 2$ bits), coarsely capturing the essence of the moderately decaying and low singular components of $\mathbf{W}$.
- Choose quantizers such that $\mathrm{B_Q} = 2$ bits, $\mathrm{B_L} = \mathrm{B_R} = 4$ bits. For an LLM weight matrix with $n = d = 4096$, choosing rank $k = 64$ implies 2.125 bits per entry.

Our Algorithm: CALDERA

Calibration Aware Low-Precision and Low-Rank Decomposition

$$\min_{\mathbf{Q},\mathbf{L},\mathbf{R}} \|(\mathbf{Q}+\mathbf{LR}-\mathbf{W})\mathbf{X}^\top\|_{\mathrm{F}}^2 \quad \text{subject to } \mathbf{Q},\mathbf{L},\mathbf{R} \text{ using } \mathrm{B_Q}, \mathrm{B_L}, \text{and } \mathrm{B_R} \text{ bits respectively.}$$

- CALDERA: **C**alibration **A**ware **L**ow-Precision **DE**composition with Low-**R**ank **A**daptation.
- Our algorithm: Alternately update $\mathbf{Q}$ and $(\mathbf{L},\mathbf{R})$.
  - Initialize $t \leftarrow 0$, $\mathbf{L}_0 \leftarrow \mathbf{0}$, $\mathbf{R}_0 \leftarrow \mathbf{0}$.
  - Step 1: $\mathbf{Q}_{t+1} \leftarrow \textsc{Quantize}(\mathbf{W}-\mathbf{L}_t\mathbf{R}_t)$ using $\mathrm{B_Q}$ bits.
    Solve $\min_{\mathbf{Q}}\|(\mathbf{Q}-\mathbf{L}_t\mathbf{R}_t-\mathbf{W})\mathbf{X}^\top\|_{\mathrm{F}}^2$ using LDLQ quantizer [Chee et al., NeurIPS '23].
  - Step 2: $\mathbf{L}_{t+1},\mathbf{R}_{t+1} \leftarrow \textsc{LPLRFactorize}(\mathbf{W}-\mathbf{Q}_{t+1},k)$, where $(\mathbf{L},\mathbf{R})$ use $(\mathrm{B_L},\mathrm{B_R})$ bits.
    Solve $\min_{\mathbf{L},\mathbf{R}}\|(\mathbf{Q}_t-\mathbf{LR}-\mathbf{W})\mathbf{X}^\top\|_{\mathrm{F}}^2$ (submodule described in next slide).
  - Iterate between Step 1 and Step 2 for a maximum number of iterations.

Low-Precision Low-Rank (LPLR) Factorize submodule

- Rank-constrained regression (RCR): $\min_{\text{rank}(\mathbf{Z}) \le k} \|\mathbf{XZ} - \mathbf{Y}\|_F^2$ is a non-convex problem that can be solved to global optimality in closed form [Xiang et al., KDD '12].
- LPLRFactorize solves RCR subject to quantization constraints, i.e., $\min_{\mathbf{L},\mathbf{R}} \|(\mathbf{LR} - \mathbf{A})\mathbf{X}^\top\|_F^2$, where $(\mathbf{L}, \mathbf{R})$ are constrained to $(\mathrm{B_L}, \mathrm{B_R})$ bits.
- For fixed $\mathbf{A}$, run an inner loop alternately update $\mathbf{L}$ and $\mathbf{R}$.
    - Initialize $(\mathbf{L}_0, \mathbf{R}_0)$ from the RCR solution.
    - Step 1: $\mathbf{L}_i = \textsc{Quantize}\left(\arg\min_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \|(\mathbf{Z}\mathbf{R}_i - \mathbf{A})\mathbf{X}^\top\|_F^2\right)$.
    - Step 2: $\mathbf{R}_i = \textsc{Quantize}\left(\arg\min_{\mathbf{Z} \in \mathbb{R}^{k \times d}} \|(\mathbf{L}_i\mathbf{Z} - \mathbf{A})\mathbf{X}^\top\|_F^2\right)$.
    - Iterate between Step 1 and Step 2 for a maximum number of inner iterations.
- *Note*: The solutions of minimization problems in steps 1 and 2 above are available in closed form.

Compressing LLaMa family of LLMs

- Results on different sizes of LLaMa models (without finetuning):[†]

Method	Rank	Avg Bits	Wiki2 ↓	C4 ↓	Wino ↑	RTE ↑	PiQA ↑	ArcE ↑	ArcC ↑
CALDERA (7B)	256	2.4	6.19	8.14	66.0	60.6	75.6	63.6	34.0
QuIP# (7B, No FT)	0	2	8.23	10.8	61.7	57.8	69.6	61.2	29.9
CALDERA (13B)	256	2.32	5.41	7.21	66.9	62.1	76.2	70.3	40.4
QuIP# (13B, No FT)	0	2	6.06	8.07	63.6	54.5	74.2	68.7	36.2
CALDERA (70B)	256	2.2	3.98	5.76	77.6	71.5	79.8	79.5	47.4
QuIP# (70B, No FT)	0	2	4.16	6.01	74.2	70.0	78.8	77.9	48.6

- Can finetune randomized Hadamard transform (RHT) parameters for improved results:

Method	Rank	Avg Bits	Wiki2 ↓	C4 ↓	Wino ↑	RTE ↑	PiQA ↑	ArcE ↑	ArcC ↑
CALDERA (7B)	256	2.4	5.84	7.75	65.7	60.6	76.5	64.6	35.9
QuIP#*	0	2	6.58	8.62	64.4	53.4	75.0	64.8	34.0

- Low-rank factors can be (optionally) fine-tuned via LoRA to boost performance on specific tasks.

Method	Rank	RHT FT	Avg Bits**	Wiki2 ↓	RTE ↑	Wino ↑
CALDERA (7B)	128	Yes	2.5	5.77	84.12	85.00
CALDERA (7B)	256	Yes	2.7	5.55	86.28	84.93

[†] E8 lattice quantization with indices packed as INT64 data type.
[*] Only end-to-end RHT finetuning, and not layer-by-layer finetuning. [**] The top 64 components of $\mathbf{L}$ and $\mathbf{R}$ are in BF16.

Summary

- We propose CALDERA for compressing an LLM in the regime of 2 to 2.5 bits per parameter, with the goal of reducing the accuracy gap to uncompressed models.
- CALDERA provides a unified framework that jointly optimizes the backbone $\mathbf{Q}$ and the low-rank factors $\mathbf{LR}$ – providing the flexibility to represent them in different precisions.
- We provide rigorous theoretical guarantees on the approximation error of CALDERA, provably showing that it is better compared to rank-agnostic compression algorithms.
- Our CALDERA decomposition can be used with other strategies like randomized Hadamard transform fine-tuning [QuIP#], Low-Rank adaptation, etc.
- Auto-regressive generation throughput for the 2 to 2.5 bit-quantized model is higher than unquantized.

Thank you!

Reach out for questions or discussions

Poster Session:
The 12 Dec 11 a.m. PST — 2 p.m. PST
https://nips.cc/virtual/2024/poster/93805

Paper: https://arxiv.org/abs/2405.18886
GitHub: https://github.com/pilancilab/caldera