Matrix Compression via Randomized Low Rank and Low Precision Factorization

Rajarshi Saha, Varun Srivastava, Mert Pilanci

Stanford ENGINEERING
Electrical Engineering

NEURAL INFORMATION PROCESSING SYSTEMS

Dec 10 - Dec 16, 2023
New Orleans

Matrix Compression

- Matrices are ubiquitous – can involve billions of elements making their storage and processing quite demanding in terms of computational resources and memory usage.

  Eg.: Vector databases, Kernel matrices, LLM weight matrices, etc.

Matrix Compression

- Matrices are ubiquitous – can involve billions of elements making their storage and processing quite demanding in terms of computational resources and memory usage.

  Eg.: Vector databases, Kernel matrices, LLM weight matrices, etc.

- Several real-world matrices exhibit **approximately low-rank structure** due to inherent redundancy or patterns. [Udell & Townsend, 2019]

Matrix Compression

- Matrices are ubiquitous – can involve billions of elements making their storage and processing quite demanding in terms of computational resources and memory usage.

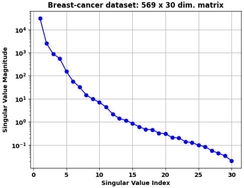
  Eg.: Vector databases, Kernel matrices, LLM weight matrices, etc.

- Several real-world matrices exhibit **approximately low-rank structure** due to inherent redundancy or patterns. [Udell & Townsend, 2019]

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

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

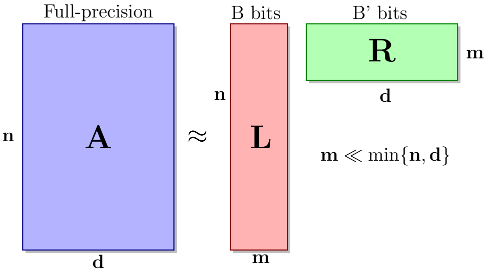
  where $\{\sigma_i\}$ are the singular values, and $\mathbf{u}_i \in \mathbb{R}^n$, $\mathbf{v}_i \in \mathbb{R}^d$ are singular vectors.



- Question: How to compress matrices via **dimensionality reduction** and **quantization**?

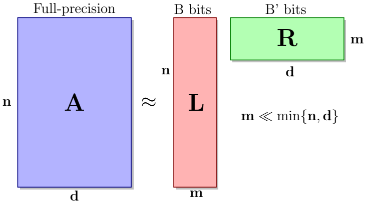
- **Our solution uses Randomized Embeddings.**

Matrix Compression: Low-precision and Low-Rank



- We obtain a **randomized factorization**, $\mathbf{A} \approx \mathbf{LR}$, where the entries of left factor $(\mathbf{L})$ and right factor $(\mathbf{R})$ are quantized with B and B' bits per entry respectively.
- Total bit requirement is $mn\mathrm{B} + md\mathrm{B}'$.
- By tuning sketch-size m we can ensure compression while letting B and B' to take values allowed by current hardware-primitives, e.g., 4-bits, 8-bits, etc.
- Low-precision computations also have **low latency**: Computing $\mathbf{Ax}$ versus $\mathbf{L}(\mathbf{Rx})$.

Matrix Compression: Low-precision and Low-Rank



Our **LPLR algorithm**:

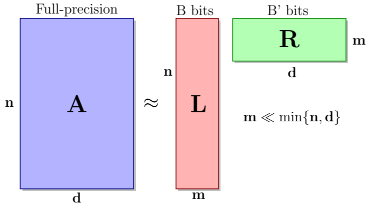
- Computes randomized rangefinder $\mathbf{AS}$, and quantizing it.
- Computes approximate projection of the columns of $\mathbf{A}$ onto this quantized basis.

Algorithm 1: **LPLR**: Randomized Low-Precision Low-Rank factorization

Input : Matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, sketch size m, Quantizers Q, Q' with dynamic ranges $\mathrm{R_Q}$, $\mathrm{R_{Q'}}$ and bit-budgets B, B' respectively.
Output : Factorization: $\mathbf{LR}$ where $\mathbf{L} \in \mathbb{R}^{n \times m}$, $\mathbf{R} \in \mathbb{R}^{m \times d}$

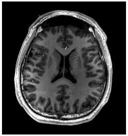
Sample a Gaussian sketching matrix $\mathbf{S} \in \mathbb{R}^{d \times m}$ with entries $S_{ij} \sim \mathcal{N}\left(0, \frac{1}{m}\right)$.
Compute an approximate basis of column space of $\mathbf{A}$ by forming the sketch: $\mathbf{AS}$.
Quantize the approximate basis with Q to get $\mathrm{Q}(\mathbf{AS})$.
Find $\mathbf{W}^* = \arg\min_{\mathbf{W}} \|\mathrm{Q}(\mathbf{AS})\mathbf{W} - \mathbf{A}\|_F^2$.
Quantize $\mathbf{W}^*$ using quantizer Q' to get $\mathrm{Q}'(\mathbf{W}^*)$.
return *Low-rank and low-precision approximation* $\mathbf{LR}$ *where* $\mathbf{L} = \mathrm{Q}(\mathbf{AS})$, $\mathbf{R} = \mathrm{Q}'(\mathbf{W}^*)$.

Matrix Compression: Low-precision and Low-Rank



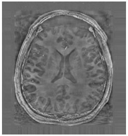
- We obtain a factorization, $\mathbf{A} \approx \mathbf{LR}$, where the entries of $\mathbf{L}$ and $\mathbf{R}$ are quantized with B and B' bits per entry respectively.
- (A popular benchmark) **Naive quantization**: Quantize each entry of $\mathbf{A} \in \mathbb{R}^{n \times d}$ uniformly with a $\mathrm{B_{nq}}$ – bit quantizer.
- Compression ratio with respect to naive quantization is $\frac{mn\mathrm{B} + md\mathrm{B}'}{nd\mathrm{B_{nq}}}$.
- By tuning sketch-size m we can ensure compression ratio ≤ 1 for $\mathrm{B_{nq}} = 1$, while letting B and B' to take values allowed by current hardware-primitives, e.g., 4-bits, 8-bits, etc.
- **Direct-SVD** quant. benchmark: Compute the best rank-k approximation $(\mathbf{U\Sigma})_k \mathbf{V}_k^\top$ by retaining the top-k singular vectors, and subsequently quantize: $\mathbf{A} \approx \mathrm{Q}((\mathbf{U\Sigma})_k)\mathrm{Q}'(\mathbf{V}_k^\top)$.

Image compression

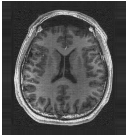


Original

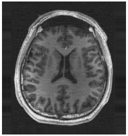
Naive



DSVD

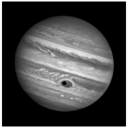


LPLR (ours)



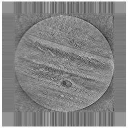
LSVD (ours)

Compressing a brain MRI. $\mathrm{B} = 4$, $\mathrm{B}' = 8$, $\mathrm{B}_{\mathrm{nq}} = 1, m = 124$, $n = 1534$ $d = 1433$

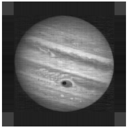


Original

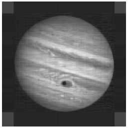
Naive



DSVD



LPLR (ours)



LSVD (ours)

Compression of a Jupiter image showing its Great Red Spot and Ganymede's shadow (NASA/ESA Hubble Space Telescope). $\mathrm{B} = 2$, $\mathrm{B}' = 8$, $\mathrm{B}_{\mathrm{nq}} = 1, m = 110$. Orig. image dim.: 1102×1102

Approximate nearest neighbor search

- For a given data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and a query $\mathbf{x} \in \mathbb{R}^d$, retrieve

$$i^* = \mathrm{argmax}_{i \in [n]} (\mathbf{A}\mathbf{x})_i \;\approx\; \mathrm{argmax}_{i \in [n]} (\mathbf{L}\mathbf{R}\mathbf{x})_i$$

- Applications: Semantic search over vector databases (music recommendation), In-context learning for LLMs, etc.

Table 5: **CIFAR100 embeddings generated by MobileNetV3 with an unquantized accuracy and F1 score** 76%:Results on LPLR and LPLR-SVD with $\mathrm{B} = \mathrm{B}' = 8$ bits

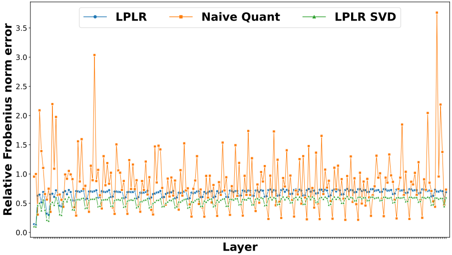
	Frobenius Norm Error				Accuracy (%)				Weighted F1 Score (%)			
B_{nq}	**LPLR**	**LSVD**	DSVD	NQ	**LPLR**	**LSVD**	DSVD	NQ	**LPLR**	**LSVD**	DSVD	NQ
1	**1.04**	1.08	1.09	6.75	79	**82**	**82**	1	79	**82**	**82**	0
2	**1.08**	1.1	1.12	2.18	**80**	**80**	**80**	1.7	**80**	**80**	**80**	1.3
4	**1.11**	1.12	1.14	1.17	**79**	78	77	75	**79**	78	78	75

Table 6: **IMDB embeddings generated by BERT with an unquantized accuracy and F1 score** 75% **and** 74% **respectively**: Results on LPLR and LPLR-SVD with $\mathrm{B} = \mathrm{B}' = 8$ bits

	Frobenius Norm Error				Accuracy (%)				Weighted F1 Score (%)			
B_{nq}	**LPLR**	**LSVD**	DSVD	NQ	**LPLR**	**LSVD**	DSVD	NQ	**LPLR**	**LSVD**	DSVD	NQ
1	0.313	**0.241**	0.229	6.63	73	74	**75**	50	74	74	**75**	33
2	0.235	0.178	**0.161**	1.016	**74**	**74**	**74**	50	**74**	**74**	**74**	50
4	0.148	0.122	**0.098**	0.417	**75**	74	**75**	73	74	74	**75**	73

Compressing weight matrices of LLMs

- **LlaMa 7b** [Touvron et. al, 2023]: An LLM with several layers (difficult to deploy on GPUs)



Comparison of LPLR and LPLR-SVD on LlaMa weight matrices with $\mathrm{B} = \mathrm{B}' = 8$ bits, $\mathrm{B}_{\mathrm{nq}} = 4$ bits, ordered by the original sequence of layers on the "Layer" – axis. We observe consistently better Frobenius norm error using LPLR and LPLR-SVD, with the exception of specific layers which lend themselves to naive quantization.

$\mathrm{B} = \mathrm{B}' = 8$ bits, $\mathrm{B}_{\mathrm{nq}} = 4$ bits			
Metric	**LPLR**	**LPLR-SVD**	Naive Quant.
Mean	0.672	**0.537**	0.836
Std Dev	0.080	**0.079**	0.470

Average relative Frobenius norm error on LlaMa weight matrices

Theoretical analysis

- Approximation error upper bounds on the Frobenius norm $\|\mathbf{LR} - \mathbf{A}\|_{\mathrm{F}}^2$.
- Bit requirement: How many bits are required per matrix coordinate to achieve the corresponding approximation error?
- Computation requirement: No. of floating point multiplications of the rate determining step. $\mathrm{O}(ndm)$ for **LPLR** vs. $\mathrm{O}(nd^2)$ for direct-SVD quant.
- Properties of randomized embeddings useful for LPLR factorization:
  - **Subspace approximation**: For approximately low-rank matrices $\mathbf{A} \in \mathbb{R}^{n \times d}$, randomly sketching the columns, i.e., $\mathbf{AS} \in \mathbb{R}^{n \times m}$ constitutes a basis for $\mathrm{range}(\mathbf{A})$ with high probability.

    [Halko et. al, 2011; Witten & Candes, 2015; Tropp et. al, 2017, ...]
  - **Democratic equalization**: $\|\mathbf{AS}\|_{\max} \triangleq \max_{i,j} |A_{ij}|$ is "small" with high probability.

    [Charikar, 2002; Boufounos & Baraniuk, 2008; Plan & Vershynin, 2014, Lyubarskii & Vershynin, 2006; Studer et. al. 2015, ...]

Details in paper

Conclusions

1. Randomized-embedding based matrix compression: Low precision and low rank representations.
2. Computationally efficient: $\mathrm{O}(ndm)$, $m \ll \min\{n, d\}$ for randomized-embedding based LPLR vs. $\mathrm{O}(nd^2)$ for direct-SVD based methods.
3. Sketch size m is a tunable knob. Allows flexible compression ratios that achieve parity with (aggressive) quantization as low as a *single* bit (using current hardware primitives).
4. Provably better approximation error guarantees (Details in paper).
5. Applications in compressing datasets, neural network weights, approximate nearest neighbors, etc.

Thank you!

Reach out for questions or discussions:
rajsaha@stanford.edu

Poster Session:
Tue 12 Dec 10:45 a.m. CST — 12:45 p.m. CST, Great Hall & Hall B1+B2 #1824
https://neurips.cc/virtual/2023/poster/70291

Paper: https://openreview.net/forum?id=rxsCTtkqA9
GitHub: https://github.com/pilancilab/matrix-compressor