BERKELEY LAB

Accelerating Multilinear Maps and Structured Sparse Tensor Kernels

EECS Department Dissertation Talk

July 21, 2025

Vivek Bharadwaj

Committee in Charge
Professor James Demmel, co-chair
Adj. Associate Professor Aydın Buluç, co-chair
Professor Katherine Yelick
Assistant Professor Michael Lindsey

Multilinear Algebra in the Exascale Era

- Linear algebra drives progress in scientific computing and machine learning.
- LLNL El Capitan: 1.7×10^{18} FP64 FLOPs on LINPACK, more in lower precision.
- Growing number of applications rely on *multilinear algebra*: much less studied / optimized. Let's survey three such applications.

Symmetry-Preserving Chemistry Models

- Scientists today can use **neural networks** to quickly approximate atomic forces, predict properties of molecular systems.
- State-of-art models even bake physical laws, like **symmetry**, into network structure.
- Accurate, high throughput simulation could revolutionize material discovery and drug design.

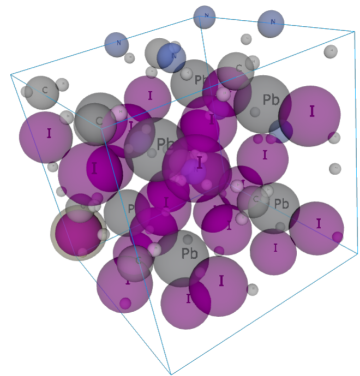
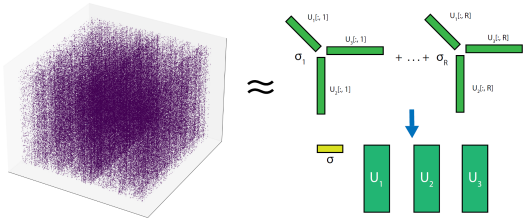


Figure: Perovskite crystal visualized with x3dase.

Multidimensional Network Analysis



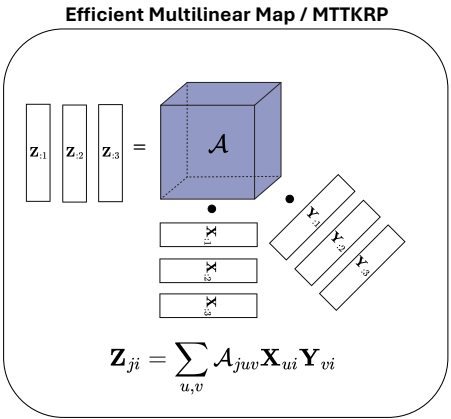
- Consider a large, sparse array containing, e.g., e-commerce review data, Reddit word usage, etc.
- We have mathematical tools to cluster / analyze quantities of interest using **only the location and values** of nonzero elements by generalizing matrix PCA.

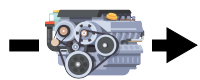
Inverse Problems and Tomography

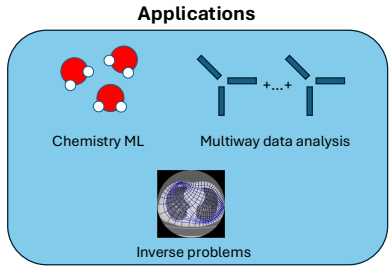
We can image interior of biological tissue by applying weak currents, measuring voltages on a boundary, solving linear least-squares problems [Che+20].

Figure: EIT demo on orange (CC2.0 Wikimedia, John Cummings).

Multilinear Maps: The Common Pattern







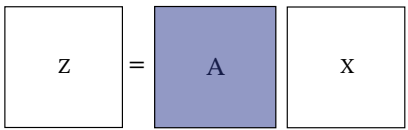
Warm-up: Linear Maps

- Consider $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ satisfying

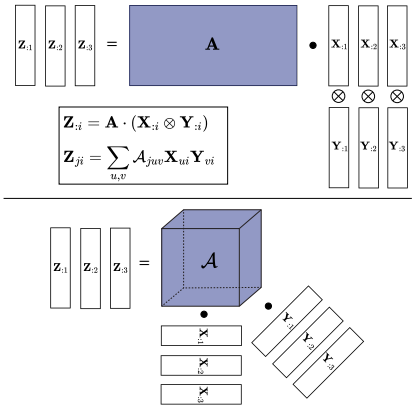
$$\boldsymbol{f}(\alpha \boldsymbol{x}) = \alpha \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{f}(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{f}(\boldsymbol{y})$$

  What linear algebraic primitive lets us evaluate such maps efficiently?

- Answer: Matrix-vector multiply for one input, matrix-matrix multiply enables high performance for multiple inputs: $\boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{X}, \boldsymbol{A} \in \mathbb{R}^{m \times n}$.



Multilinear Maps and the MTTKRP



- Consider a multilinear function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$. What fundamental kernels allow efficient evaluation?

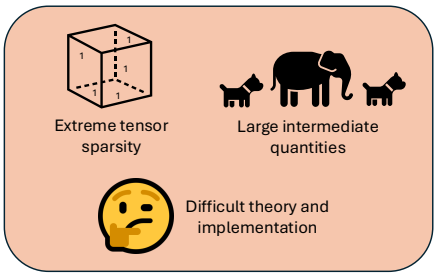
- Answer: matrix-vector multiply by a **Kronecker product**:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{A}(\boldsymbol{x} \otimes \boldsymbol{y}), \boldsymbol{A} \in \mathbb{R}^{k \times mn},$$

  matrix-matrix multiplication by a **Khatri-Rao product** for a batch:

$$\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{Y}) = \boldsymbol{A}(\boldsymbol{X} \odot \boldsymbol{Y}).$$

Challenges and Opportunities



- Sparse tensors (e.g. Reddit-2015 [Smi+17]) exhibit nonzero fraction as low as 4×10^{-10}; curse of dimensionality.
- Kronecker product height is exponential in input dimension; costly to materialize.
- Existing algorithms / software target matrices, not tensors.

We accelerate the multilinear map in two important applications, offering OOM speedups via kernel engineering and asymptotically faster approximation algorithms.

Research Papers Covered Today

In this talk, we explore multiple methods to accelerate the **batched multilinear map**:

1. **Optimizing Chemistry Foundation Models:** An Efficient Sparse Kernel Generator for O(3)-Equivariant Deep Networks. ACDA [Bha+25].

2. **Accelerated Randomized Tensor Decomposition:** Fast Exact Leverage Score Sampling from Khatri-Rao Products. NeurIPS [Bha+23].

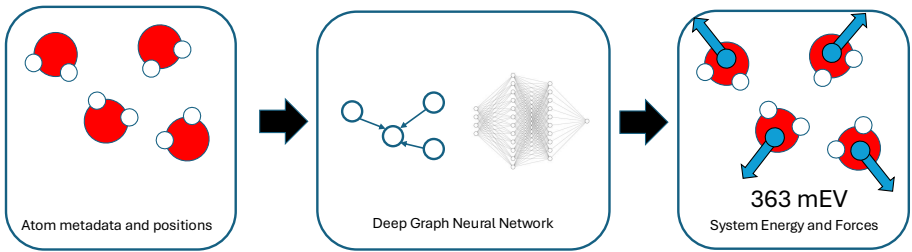
3. Distributed-Memory Randomized Algorithms for Sparse Tensor CP Decomposition. SPAA [Bha+24a].

4. Efficient Leverage Score Sampling for Tensor Trains. NeurIPS [Bha+24b].

5. Distributed-Memory Sparse Kernels for Machine Learning. IPDPS [BBD22].

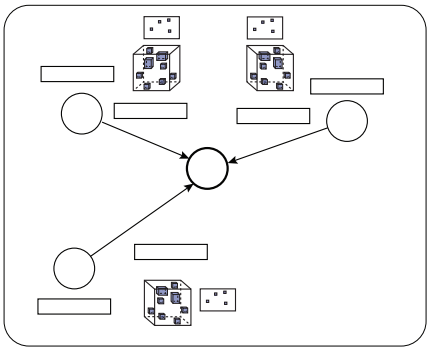
Part 1: Clebsch-Gordon Tensor Products

Geometric Deep Learning in Atomic Simulation



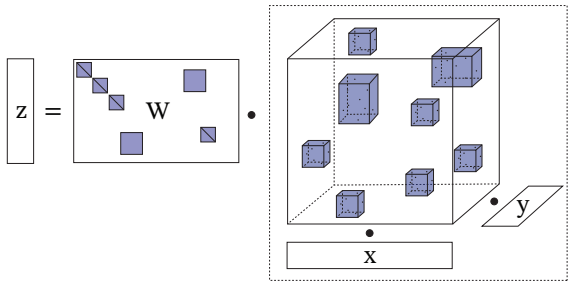
- Geometric deep learning achieves SOTA performance for fast interatomic potential calculation. Key primitive is the **message-passing graph neural network**.
- Input: atom metadata and positions. Output: system energy, atomic forces.

Respecting Physical Symmetries



- Message-passing GNNs generate messages for each node and edge.
- O(3)-Equivariance: if the input coordinate system rotates, the output energy stays **the same** and the predicted forces **rotate compatibly**.
- Need to combine node, edge features in a highly structured, prescriptive manner.

The Clebsch-Gordon Tensor Product



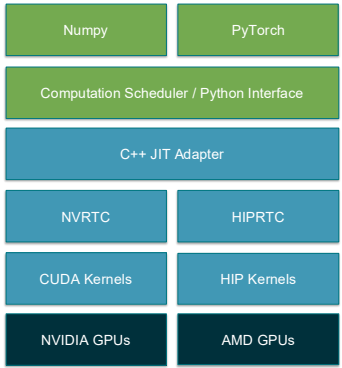
$$\boldsymbol{z} = \boldsymbol{W} \cdot \boldsymbol{P} \cdot (\boldsymbol{x} \otimes \boldsymbol{y}) = \boldsymbol{W} \sum_{i=1,j=1}^{m,n} \boldsymbol{x}\,[i]\,\boldsymbol{y}\,[j]\,\mathcal{P}\,[ij:]$$

Kernel typically executes on a large batch of $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{W})$ inputs ($B \approx 10^5 - 10^7$).

Our Contributions [Bha+25]

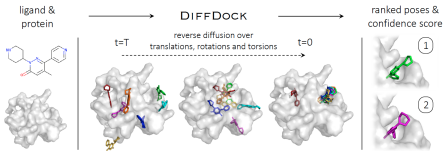
We introduce *OpenEquivariance*, a fast kernel generator for CG tensor products with up to **10x speedup** over the popular e3nn package, on par with NVIDIA cuE (joint work with Austin Glover).

- Exploits sparse structure in the multilinear map tensor via JIT, register caching, loop unrolling.
- Provides FlashAttention-style [Dao+22] backward pass, novel identities for higher gradients.
- Fuses CG tensor product with graph convolution, saving orders of magnitude of memory.

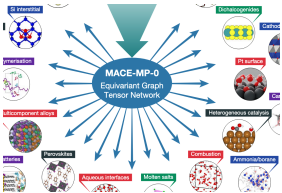


Applications of OpenEquivariance

(a) Nequip [Bat+22]



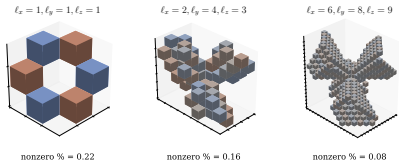
(b) DiffDock [Cor+23]

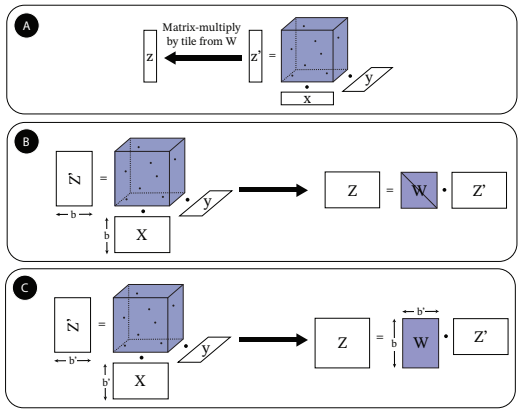


(c) MACE-MP0 [Bat+24]

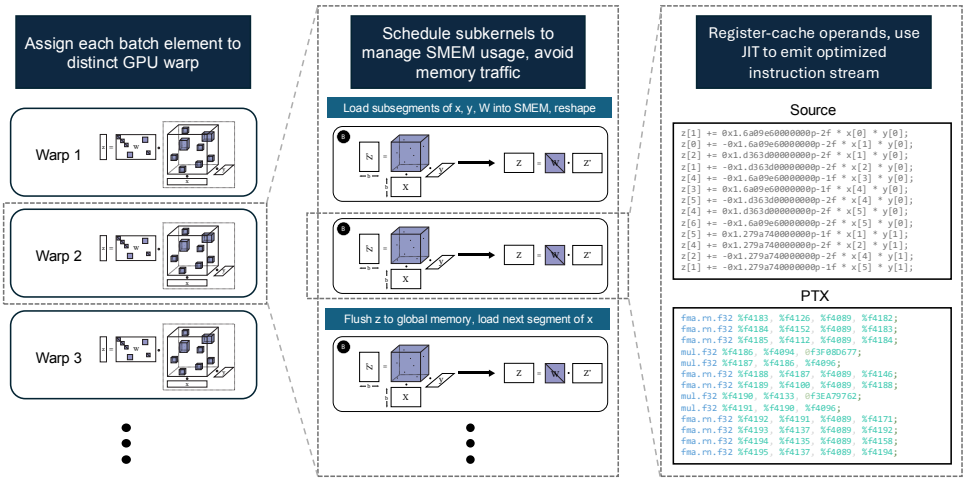
Subkernels of the CG Tensor Product

- Many nonzero blocks repeated in the sparse tensor (known at model compile-time).
- Operation splits into many smaller operations (subkernels)





A Roadmap to Efficient CG Kernels



Warp-Level Forward Algorithm

Algorithm Subkernel C Warp-Level Algorithm

Require: $\boldsymbol{X} \in \mathbb{R}^{b' \times (2\ell_x+1)}, \boldsymbol{y} \in \mathbb{R}^{(2\ell_y+1)}, \boldsymbol{W} \in \mathbb{R}^{b \times b'}$
Require: Sparse tensor $\mathcal{P}^{(\ell_x,\ell_y,\ell_z)}$ for subkernel
  for $t = 1 \ldots b'$ **do** ▷ Parallel over threads
    Load $\boldsymbol{x}_{\text{reg}} = \boldsymbol{X}[t,:], \boldsymbol{y}_{\text{reg}} = \boldsymbol{y}$
    Initialize $\boldsymbol{z}_{\text{reg}} \in \mathbb{R}^{2\ell_z+1}$ to 0.

    for $(i,j,k,v) \in \text{nz}(\mathcal{P}^{(\ell_x,\ell_y,\ell_z)})$ **do** ▷ Unroll via JIT
      $\boldsymbol{z}_{\text{reg}}[k] \mathrel{+}= v \cdot \boldsymbol{x}_{\text{reg}}[i] \cdot \boldsymbol{y}_{\text{reg}}[j]$

    Store $\boldsymbol{Z}'[t,:] = \boldsymbol{z}_{\text{reg}}$, compute $\boldsymbol{Z} \mathrel{+}= \boldsymbol{W} \cdot \boldsymbol{Z}'$.

- Gradient calculation (backward pass) is similar, but requires 3 update equations instead of 1.

Optimizing Model Force Predictions

- The derivative of predicted energy $E(\boldsymbol{R}, \boldsymbol{W})$ w.r.t. atomic positions $\boldsymbol{R}$ yields atomic forces. Want to solve

$$\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{R}, \boldsymbol{W}) = \min_{\boldsymbol{W}} \|\boldsymbol{F}_{\mathrm{pr}}(\boldsymbol{R}, \boldsymbol{W}) - \boldsymbol{F}_{\mathrm{gt}}(\boldsymbol{R})\|_F^2$$

- To minimize difference between predicted forces and ground truth, need *second partial derivative of energy*:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} &= 2 \cdot \mathrm{vec}(\boldsymbol{F}_{\mathrm{pr}}(\boldsymbol{R}, \boldsymbol{W}) - \boldsymbol{F}_{\mathrm{gt}}(\boldsymbol{R}))^\top \frac{\partial \boldsymbol{F}_{\mathrm{pr}}}{\partial \boldsymbol{W}} \\ &= -2 \cdot \mathrm{vec}(\boldsymbol{F}_{\mathrm{pr}}(\boldsymbol{R}, \boldsymbol{W}) - \boldsymbol{F}_{\mathrm{gt}}(\boldsymbol{R}))^\top \frac{\partial^2 E}{\partial \boldsymbol{R} \partial \boldsymbol{W}}\end{aligned}$$

Novel Identities for Second Partial Derivatives

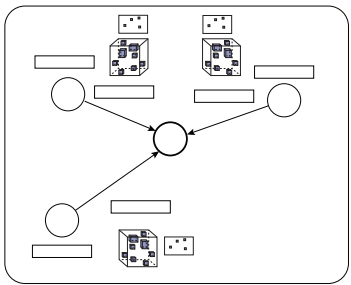
- PyTorch gladly takes the second partial derivative... if you supply a "double-backward" pass for the CG tensor product.

- **Key**: No additional kernel engineering required! Double-backward can be expressed as a linear combination of forward / backward calls:

$$
\begin{aligned}
\text{op1} &= \text{backward}(\partial\mathcal{L}/\partial\boldsymbol{a}, \partial\mathcal{L}/\partial\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{g}_z) \\
\text{op2} &= \text{backward}(\boldsymbol{x}, \boldsymbol{y}, \partial\mathcal{L}/\partial\boldsymbol{C}, \boldsymbol{g}_z) \\
\text{op3} &= \text{TP}(\partial\mathcal{L}/\partial\boldsymbol{a}, \boldsymbol{y}, \boldsymbol{W}) \\
\text{op4} &= \text{backward}(\partial\mathcal{L}/\partial\boldsymbol{a}, \boldsymbol{y}, \boldsymbol{W}, \boldsymbol{g}_z) \\
\text{op5} &= \text{backward}(\boldsymbol{x}, \partial\mathcal{L}/\partial\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{g}_z) \\
\text{op6} &= \text{TP}(\boldsymbol{x}, \partial\mathcal{L}/\partial\boldsymbol{b}, \boldsymbol{W}) \\
\text{op7} &= \text{TP}(\boldsymbol{x}, \boldsymbol{y}, \partial\mathcal{L}/\partial\boldsymbol{C})
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
\partial\mathcal{L}/\partial\boldsymbol{x} &= \text{op1}\,[1] + \text{op2}\,[1] \\
\partial\mathcal{L}/\partial\boldsymbol{y} &= \text{op1}\,[2] + \text{op2}\,[2] \\
\partial\mathcal{L}/\partial\boldsymbol{W} &= \text{op4}\,[3] + \text{op5}\,[3] \\
\partial\mathcal{L}/\partial\boldsymbol{g}_z &= \text{op3} + \text{op6} + \text{op7}
\end{aligned}
$$

Kernel Fusion

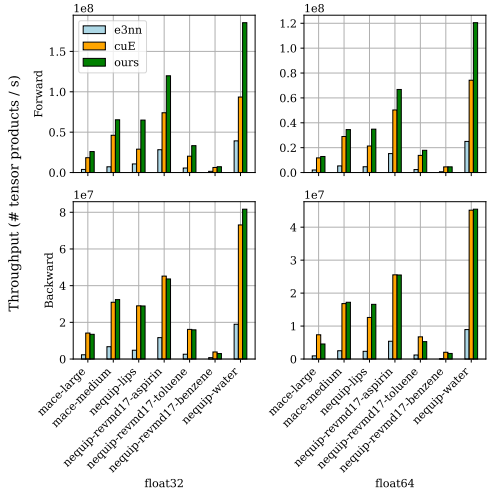
- Graph convolution has SpMM memory access pattern
- Fuse both kernels to save compute AND memory!



Algorithm Deterministic TP + Graph Convolution

Require: Graph $G = (V, E)$, $E[b] = (i_b, j_b)$
Require: Batch $\boldsymbol{x}_1, ..., \boldsymbol{x}_{|V|}, \boldsymbol{y}_1, ...\boldsymbol{y}_{|E|}, \boldsymbol{W}_1, ..., \boldsymbol{W}_{|E|}$
  for $\text{segment}_i \in \text{schedule}$ **do**
    $(s, t) = E[k][0], E[k][1]$
    Set $\boldsymbol{z}_{\text{acc}} = 0$ ▷ Parallel over Warps
    for $b = 1... |E|$ **do**
      Execute segment subkernel sequence
      $\boldsymbol{z}_{\text{acc}} \mathrel{+}= \boldsymbol{z}_{\text{smem}}$
      if $b = |E|$ or $s < E[b+1][0]$ **then**
        if s is first vertex processed by warp **then**
          Send $\boldsymbol{z}_{\text{acc}}$ to fixup buffer.
        else
          Store $\boldsymbol{z}_{\text{acc}}$ to global memory.
        $\boldsymbol{z}_{\text{acc}} = 0$
  Execute fixup kernel.

Throughput (Unfused, A100)

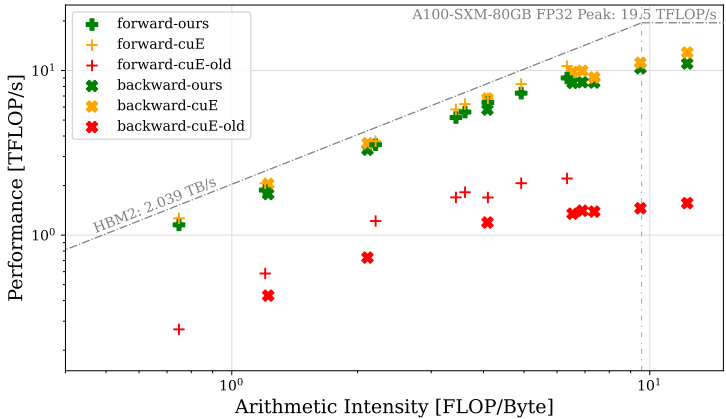


Cross-Platform Performance

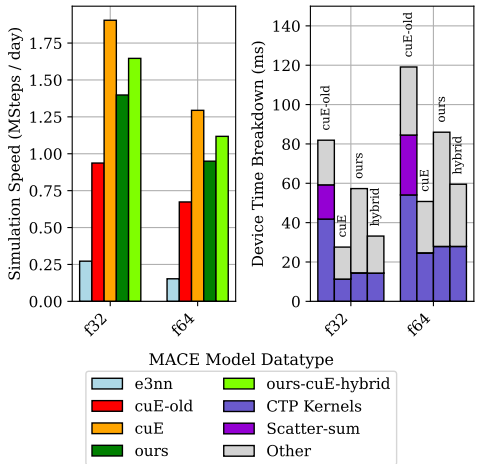
GPU	forward			backward		
	e3nn	cuE	ours	e3nn	cuE	ours
A100	13	2.8	**2.0**	21	**3.5**	3.7
A5000	29	4.2	**3.8**	42	**9.3**	11
MI250x	41	-	**3.0**	128	-	**15**

Table: MACE-large isolated tensor product runtime (ms), batch size 50K, FP32 unfused.

Roofline Analysis



Acceleration of MACE Foundation Model



Takeaways

- OpenEquivariance achieves SOTA performance, provides 5-6x end-to-end speedup for Nequip / MACE. 77 Github stars (and counting!)
- **Key Point:** Exploited structure in the tensor to engineer high performance kernels.
- **Related Work**: Graph neural networks have myriad applications; we study another useful kernel, SDDMM, in our IPDPS 2022 paper [BBD22].
- **Next**: a more general problem where we must **discover structure** in the sparse tensor.

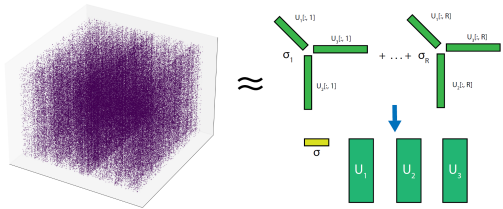
Part 2: Randomized Sparse Tensor Decomposition

Motivating Application

- **Our goal:** efficiently solve an overdetermined linear least-squares problem

$$\min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F$$

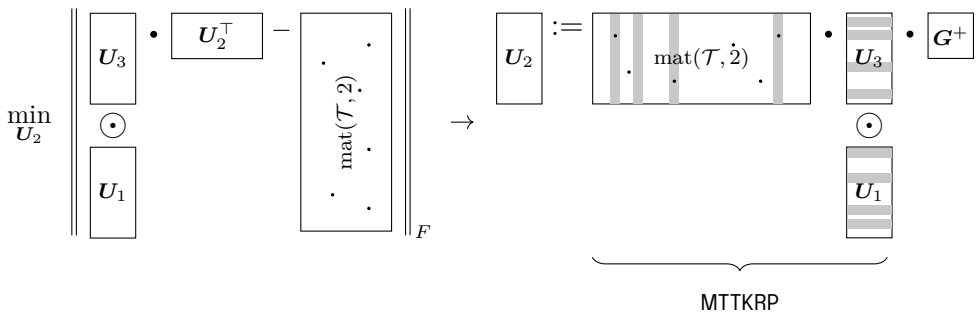
where $A = \boldsymbol{U}_1 \odot \ldots \odot \boldsymbol{U}_N$ with $\boldsymbol{U}_j \in \mathbb{R}^{I \times R}$.



- Key kernel in alternating least-squares Candecomp / PARAFAC (CP) decomposition [LK22].

ALS and the Multilinear Map

$$\min_{\boldsymbol{U}_j} \left\| \left[\bigodot_{k \neq j} \boldsymbol{U}_k \right] \cdot \boldsymbol{U}_j^\top - \mathrm{mat}(\mathcal{T}, j)^\top \right\|_F$$



Randomized Linear Least-Squares

- **Sketch & Solve**: Apply short-wide sketching matrix $\boldsymbol{S}$ to both $\boldsymbol{A}$ and $\boldsymbol{B}$, solve reduced problem

$$\min_{\tilde{\boldsymbol{X}}} \left\| \boldsymbol{S}\boldsymbol{A}\tilde{\boldsymbol{X}} - \boldsymbol{S}\boldsymbol{B} \right\|_F$$

- Want an (ε, δ) guarantee on solution quality: with high probability $(1-\delta)$,

$$\left\| \boldsymbol{A}\tilde{\boldsymbol{X}} - \boldsymbol{B} \right\|_F \leq (1+\varepsilon) \min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|$$

- In this talk: we will just **randomly sample** J rows of A and B to form a pair of much shorter matrices. Preserves structure in both input and output.

Our Contributions [Bha+23]

Method	Source	Round Complexity ($\tilde{O}$ notation)
CP-ALS	[KB09]	$N(N+I)I^{N-1}R$
CP-ARLS-LEV	[LK22]	$N(R+I)R^N/(\varepsilon\delta)$
TNS-CP	[Mal22]	$N^3IR^3/(\varepsilon\delta)$
GTNE	[MS22]	$N^2(N^{1.5}R^{3.5}/\varepsilon^3 + IR^2)/\varepsilon^2$
STS-CP	Ours	$N(NR^3 \log I + IR^2)/(\varepsilon\delta)$

- We build a data structure with runtime **logarithmic** in the height of the KRP and quadratic in R to sample from *leverage scores* of A.

- Yields the **STS-CP** algorithm: lower asymptotic runtime for randomized dense CP decomposition than recent SOTA methods (even more advantage for sparse tensors).

Intuition: Do I Need Every Row?

- Consider a univariate regression problem with 100,000 (x, y) points.
- This is a **highly-overdetermined** problem. Can pick a subset of points to perform fitting.

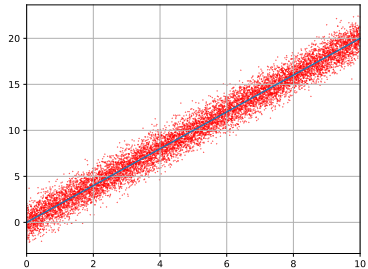


Figure: $y_i \sim x_i + \mathcal{N}(0, 1)$

Leverage Score Sampling

We will sample rows i.i.d. from $\boldsymbol{A}$ according to the *leverage score distribution* on its rows. Given **reduced SVD** $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, define

$$\ell_i = \|\boldsymbol{U}\,[i,:]\|^2 .$$

Theorem (Leverage Score Sampling Guarantees, [Mal22])

Suppose $\boldsymbol{S} \in \mathbb{R}^{J\times I}$ is a leverage-score sampling matrix for $\boldsymbol{A} \in \mathbb{R}^{I\times R}$, and define

$$\tilde{\boldsymbol{X}} := \arg\min_{\tilde{\boldsymbol{X}}} \left\| \boldsymbol{S}\boldsymbol{A}\tilde{\boldsymbol{X}} - \boldsymbol{S}\boldsymbol{B} \right\|_F$$

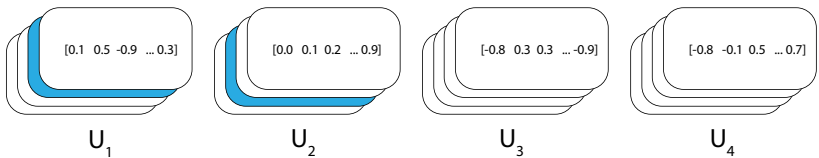
If $J \gtrsim R\max(\log(R/\delta), 1/(\varepsilon\delta))$, then with probability at least $1-\delta$,

$$\left\| \boldsymbol{A}\tilde{\boldsymbol{X}} - \boldsymbol{B} \right\|_F \leq (1+\varepsilon) \min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F .$$

Implicit Leverage Score Sampling

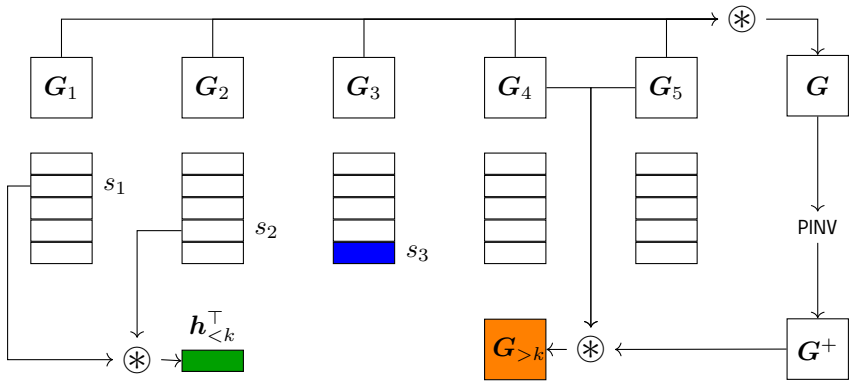
- For $I = 10^7, N = 3$, matrix $\boldsymbol{A}$ has 10^{21} rows. Can't even index rows with 64-bit integers. Instead: use identity $\ell_i = \boldsymbol{A}\,[i,:]\,(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}\,[i,:]^\top$.

- Draw a row from each of $\boldsymbol{U}_1, ..., \boldsymbol{U}_N$, return Hadamard product.



- Let $\hat{s}_j$ be a random variable for the row index drawn from $\boldsymbol{U}_j$. Assume $(\hat{s}_1, ..., \hat{s}_N)$ jointly follows leverage score distribution on $\boldsymbol{U}_1 \odot ... \odot \boldsymbol{U}_N$.

The Conditional Distribution of $\hat{s}_k$



$$p(\hat{s}_k = s_k \mid \hat{s}_{<k} = s_{<k}) \propto \langle \boldsymbol{h}_{<k}\boldsymbol{h}_{<k}^\top, \boldsymbol{U}_k[s_k, :]^\top \boldsymbol{U}_k[s_k, :], \boldsymbol{G}_{>k} \rangle$$

Key Sampling Primitive

$$\boldsymbol{q}[i] := C^{-1}\langle \boldsymbol{h}_{<k}\boldsymbol{h}_{<k}^{\top}, \boldsymbol{U}_k[i,:]^{\top}\boldsymbol{U}_k[i,:], \boldsymbol{G}_{>k}\rangle$$

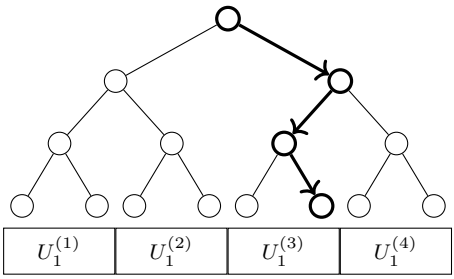
- Imagine we magically had all entries of $\boldsymbol{q}$ - how to sample? Initialize I bins, j-th has width $\boldsymbol{q}[j]$.
- Choose random real r in $[0,1]$, find "containing bin" i such that

$$\sum_{j=0}^{i-1}\boldsymbol{q}[j] < r < \sum_{j=0}^{i}\boldsymbol{q}[j]$$

Binary Tree Inversion Sampling

- Locate bin via binary search (truncated to $\log(I/R)$ levels)
- Root: branch right iff $\sum_{j=0}^{I/2} \boldsymbol{q}[j] < r$
- Level 2: branch right iff

$$\sum_{j=0}^{I/2} \boldsymbol{q}[j] + \sum_{j=I/2}^{3I/4} \boldsymbol{q}[j] < r$$



Key: Can compute summations quickly if we cache information at each node!

Caching Partial Gram Matrices

Associate internal node v to an interval of rows $[S(v)...E(v)]$.

$$\sum_{j=S(v)}^{E(v)} \boldsymbol{q}\,[j] = \sum_{j=S(v)}^{E(v)} C^{-1}\langle \boldsymbol{h}_{<k}\boldsymbol{h}_{<k}^{\top}, \boldsymbol{U}_k\,[j,:]^{\top}\,\boldsymbol{U}_k\,[j,:], \boldsymbol{G}_{>k}\rangle$$

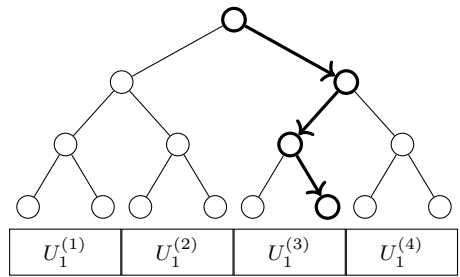
$$\ldots$$

$$:= C^{-1}\langle \boldsymbol{h}_{<k}\boldsymbol{h}_{<k}^{\top}, \boldsymbol{G}^{v}, \boldsymbol{G}_{>k}\rangle$$

Can compute $\boldsymbol{G}^v$ for ALL nodes v in time $O(IR^2)$, storage space $O(IR)$. Use BLAS-3 **syrk** calls in parallel to efficiently construct the tree.

Efficient Sampling after Caching

- At internal nodes, compute $C^{-1}\langle \boldsymbol{h}_{<k}\boldsymbol{h}_{<k}^{\top}, \boldsymbol{G}^{v}, \boldsymbol{G}_{>k}\rangle$ in $O(R^2)$ time (read normalization constant from root).
- At leaves, spend $O(R^3)$ time to compute remaining values of $\boldsymbol{q}$. Can reduce to $O(R^2 \log R)$, see paper.
- Complexity per sample: $O(NR^2 \log I)$ (summed over all tensor modes).



Geometry Preservation

Define $D(\boldsymbol{S}, \boldsymbol{A})$ of sketch $\boldsymbol{S}$ with respect to matrix $\boldsymbol{A}$ by

$$D(\boldsymbol{S}, \boldsymbol{A}) = \kappa(\boldsymbol{S}\boldsymbol{Q}) - 1$$

where $\boldsymbol{Q}$ is any orthonormal basis for the column space of $\boldsymbol{A}$. Quantifies the distance preservation property of a sketch.

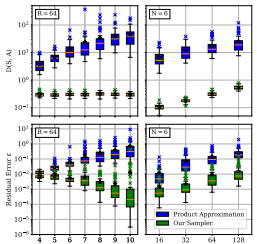


Figure: $D(\boldsymbol{S}, \boldsymbol{A})$ as a function of KRP matrix count N and column count R, $J = 65,536$. Green: our sampler. Blue: product approximation by [LK22].

Accuracy Comparison for Fixed Sample Count

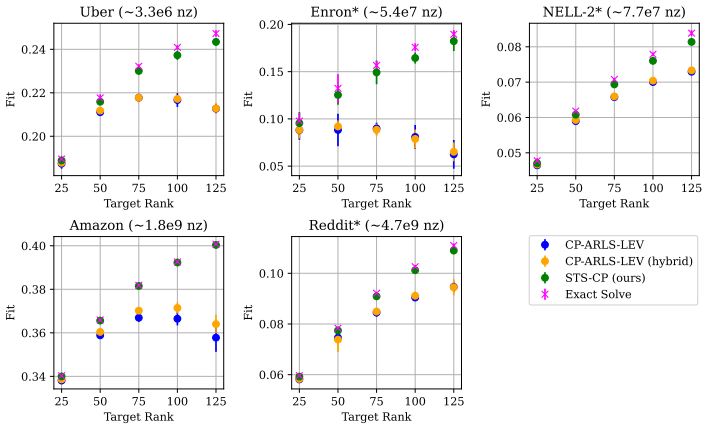


Figure: Sparse tensor ALS accuracy comparison for $J = 2^{16}$ samples, varied target ranks.

STS-CP Makes Faster Progress to Solution

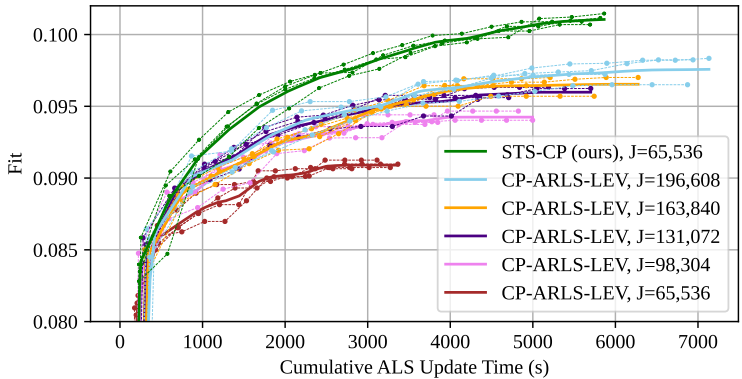


Figure: Fit vs. ALS update time, Reddit tensor, $R = 100$.

Takeaways

- Our data structure can downsample Khatri-Rao products with trillions of rows.
- Effectively preserves column-space geometry, offers OOM sample efficiency boost, 1.5-2.5x speedup over SOTA baseline methods.
- Data structures rely on common BLAS2, BLAS3 primitives, rendering them efficient.
- Compare to part 1: here, we exploit structure in the Khatri-Rao product, not the tensor.

Extension 1: Distributed Parallelism [Bha+24a]

- **SPAA 2024 Paper**: High-performance implementations of STS-CP and CP-ARLS-LEV in the distributed-memory parallel setting.
- Up to 11x speedup over SPLATT on hundreds of MPI ranks / thousands of CPU cores.
- We optimize both communication and computation in multiple stages of the solver.

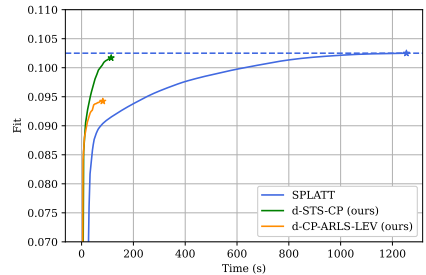
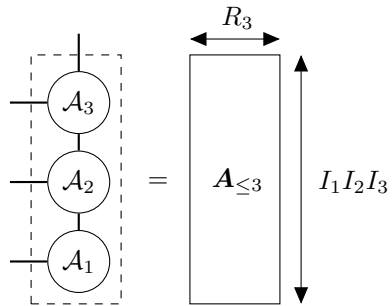


Figure: Accuracy vs. time, Reddit tensor, $R = 100$, 512 cores / 4 Perlmutter CPU nodes, 4.7B NNZ.

Extension 2: Sketched TT [Bha+24b]

- **NeurIPS 2024:** Apply theoretical machinery developed here to sketch *tensor trains* or *matrix product states*.
- Consider cores $\mathcal{A}_1, ..., \mathcal{A}_j$, let $\boldsymbol{A}_{\leq j}$ be the *matricization* of the chain.
- When cores are in a special canonical form, exploit tree DS to sample $\boldsymbol{A}_{\leq j}$ with high efficiency.



Future Work: Inverse Problems

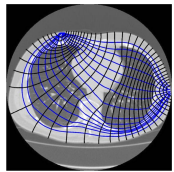


Figure: Electric field lines, human thorax (CC3.0 Wikimedia, Andy Adler).

- 2D EIT requires us to solve a linear least-squares problem of the form

$$\min_{\boldsymbol{x}} \left\| \left(\boldsymbol{A}_{11} \odot \boldsymbol{A}_{12} + \boldsymbol{A}_{21} \odot \boldsymbol{A}_{22} \right) \boldsymbol{x} - \boldsymbol{b} \right\|^2$$

  where $\boldsymbol{A}_{11}, \boldsymbol{A}_{12}, \boldsymbol{A}_{21}, \boldsymbol{A}_{22}$ depend on geometry. Must adapt leverage-based sketch.

- More interesting problem: where to place electrodes along boundary to gain maximum information? *Sensor placement problem*, solve by volume (determinant) maximization.

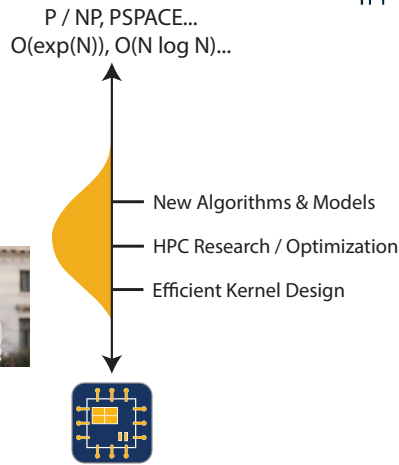
Analysis and Future Directions

The Upshot

- This presentation focused on a single operation… but we used techniques from
  - ILP-maximizing, fine-grained GPU kernel engineering
  - Randomized algorithm theory
  - Distributed-memory communication analysis
- We explored parallel computing motifs that include
  - Dense matrix multiply (warp-level)
  - Multiple sparse matrix / tensor primitives
  - Particle simulation / molecular dynamics
- We accelerated applications ranging from chemistry foundation models to multidimensional pattern mining / data analysis.
- A lot of mileage for one simple primitive!

Future Work

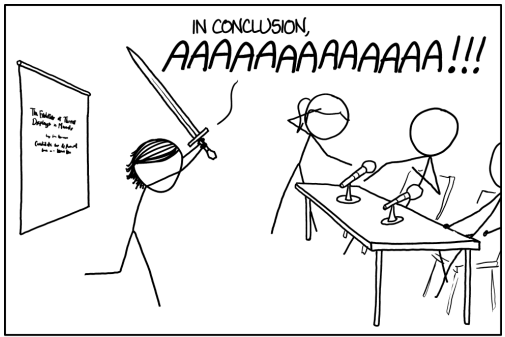
- **Up the Stack**: How do we co-design new MLIP models / algorithms to exploit accelerated primitives?
- **Down the Stack**: How do we design systems (e.g. ML compilers) to automate kernel engineering while delivering high performance?
- **Scientific / Industry Collaborations**: So much needs acceleration - got a problem? Let's talk.



Acknowledgements - Thank you to...

Everyone in Attendance!

(and your taxpayer dollars funding this research)



THE BEST THESIS DEFENSE IS A GOOD THESIS OFFENSE.

https://www.xkcd.com/1403/

My Advisers

Aydın Buluç

James Demmel

Committee, Collaborators, and Fellow Grads

Committee Members: Michael Lindsey and Katherine Yelick– thank you!

Collaborators: Russell Castro, Austin Glover, Laura Grigori, Riley Murray, Guillaume Rabusseau and Beheshteh Rakhshan, with very special thanks to Osman Malik.

Members of PASSION and BeBOP: Benjamin Brock, Alok Tripathy, Giulia Guidi, Roger Hsiao, Tianyu Liang, Gabriel Raulet, Yen-Hsiang Chang.

Supporters Over Many Years

Undergraduate / internship advisers: Chris Umans, Rose Yu, Mikhail Shapiro, and Adam Sheffer while I was at Caltech, and Federico Busato at NVIDIA!

Members of the CSGF Program: Koby Hayashi, Caleb Ju, Grace Wei, Mansi Sakarvadia, Santiago Vargas, Albert Muesalian, Gabby Jones, Julian Bellavita, Kiran Eiden, Mary Francis LaPorte, Olorundamilola Kazeem, Ethan Epperly, Katherine Keegan, and so many others.

Especially: Allison Wang, Anant Kale, Hanwen Zhang, Emily Pan, Daniel Mark. And of course, Aditi Lahiri. Thank you for everything.

My Family

La fin!

References I

[Bat+22] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. "E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials". In: *Nature Communications* 13.1 (May 2022), p. 2453. ISSN: 2041-1723. DOI: 10.1038/s41467-022-29939-5. URL: https://doi.org/10.1038/s41467-022-29939-5.

[Bat+24] Ilyes Batatia et al. *A foundation model for atomistic materials chemistry*. 2024. arXiv: 2401.00096 [physics.chem-ph]. URL: https://arxiv.org/abs/2401.00096.

[BBD22] Vivek Bharadwaj, Aydın Buluç, and James Demmel. "Distributed-Memory Sparse Kernels for Machine Learning". In: *2022 IEEE International Parallel and Distributed Processing Symposium*. Los Alamitos, CA, USA: IEEE Computer Society, June 2022, pp. 47–58. DOI: 10.1109/IPDPS53621.2022.00014.

[Bha+23] Vivek Bharadwaj, Osman Asif Malik, Riley Murray, Laura Grigori, Aydın Buluç, and James Demmel. "Fast Exact Leverage Score Sampling from Khatri-Rao Products with Applications to Tensor Decomposition". In: *Thirty-seventh Conference on Neural Information Processing Systems*. Dec. 2023.

References II

[Bha+24a] Vivek Bharadwaj, Osman Asif Malik, Riley Murray, Aydın Buluç, and James Demmel. "Distributed-Memory Randomized Algorithms for Sparse Tensor CP Decomposition". In: *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '24. Nantes, France: Association for Computing Machinery, May 2024, pp. 155–168. ISBN: 9798400704161. DOI: 10.1145/3626183.3659980. URL: https://doi.org/10.1145/3626183.3659980.

[Bha+24b] Vivek Bharadwaj, Beheshteh T. Rakhshan, Osman Asif Malik, and Guillaume Rabusseau. "Efficient Leverage Score Sampling for Tensor Train Decomposition". In: *Thirty-eighth Conference on Neural Information Processing Systems*. Dec. 2024. arXiv: 2406.02749 [cs.DS].

[Bha+25] Vivek Bharadwaj, Austin Glover, Aydın Buluç, and James Demmel. "An Efficient Sparse Kernel Generator for O(3)-Equivariant Deep Networks". In: *Proceedings of the 3rd SIAM Conference on Applied and Discrete Computational Algorithms*. July 2025. URL: https://arxiv.org/abs/2501.13986.

[Che+20] Ke Chen, Qin Li, Kit Newton, and Stephen J. Wright. "Structured Random Sketching for PDE Inverse Problems". In: *SIAM Journal on Matrix Analysis and Applications* 41.4 (2020), pp. 1742–1770. DOI: 10.1137/20M1310497.

References III

[Cor+23] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi S. Jaakkola. "DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Dao+22] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 16344–16359.

[KB09] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 2009). Publisher: Society for Industrial and Applied Mathematics, pp. 455–500. ISSN: 0036-1445. DOI: 10.1137/07070111X.

[LK22] Brett W. Larsen and Tamara G. Kolda. "Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 43.3 (2022), pp. 1488–1517.

References IV

[Mal22] Osman Asif Malik. “More Efficient Sampling for Tensor Decomposition With Worst-Case Guarantees”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 14887–14917.

[MS22] Linjian Ma and Edgar Solomonik. “Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 38980–38993.

[Smi+17] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. *FROSTT: The Formidable Repository of Open Sparse Tensors and Tools*. 2017. URL: `http://frostt.io/`.