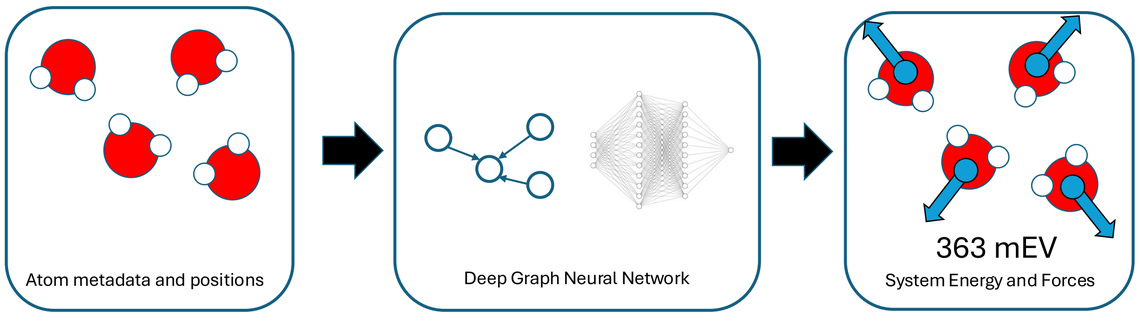
Engineering Fast Kernels for Rotation-Equivariant Deep Neural Networks

Vivek Bharadwaj *,1,2 Austin Glover *,1 Aydın Buluç 2,1 James Demmel 1

* Equal Contribution, 1 UC Berkeley, 2 Lawrence Berkeley National Lab

Atomic Simulation via Geometric Deep Learning

Motivation: Geometric deep learning on point clouds is increasingly popular in computational chemistry, materials science research, protein simulation.



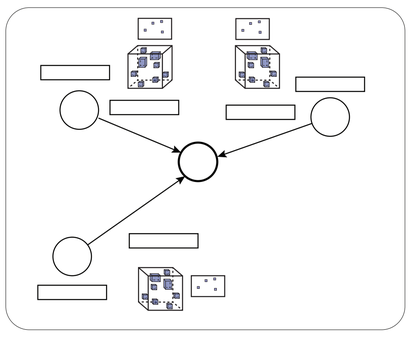
Interatomic Potential Calculation: Given a configuration of atoms in space and associated metadata, calculate total potential energy. Derivatives of potential energy yield atomic forces that feed to LAMMPS, other MD simulators.

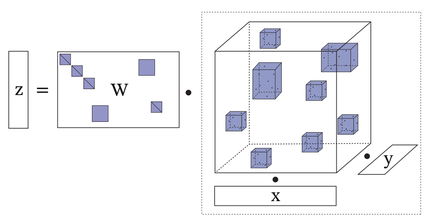
O(3)-Equivariant Graph Neural Networks

Popular model is a deep graph convolutional network that does the following:

1. Generates an embedding vector for each node and edge.
2. Combines node and edge features to produce new embeddings.
3. Sums these embeddings across the neighborhood of each node.

Key Criteria: If the input point cloud rotates about the origin, energy predictions should not change and force predictions rotate compatibly.





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

Enforcing symmetry enhances generalization / training data efficiency, but requires that we combine the node and edge features via the Clebsch-Gordon (CG) tensor product (above right). Here, $\boldsymbol{x}, \boldsymbol{y}$ are node / edge features, $\mathcal{P}$ is a block sparse tensor known at model compile-time, $\boldsymbol{W}$ contains trainable weights.

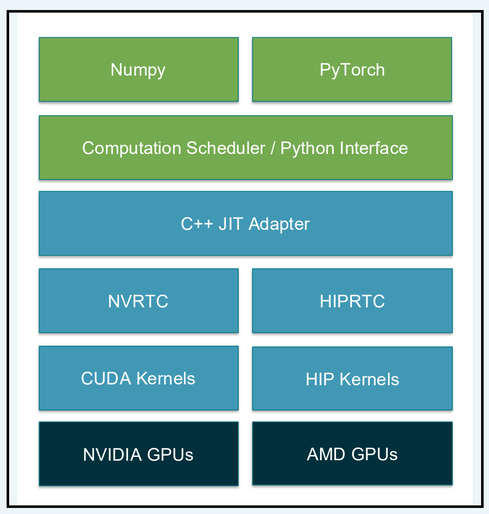
Problem: Modern GPUs are optimized for matrix-multiply and other regular calculations. How do we design efficient kernels for the CG tensor product?

OpenEquivariance: Supercharging CG Kernel Performance

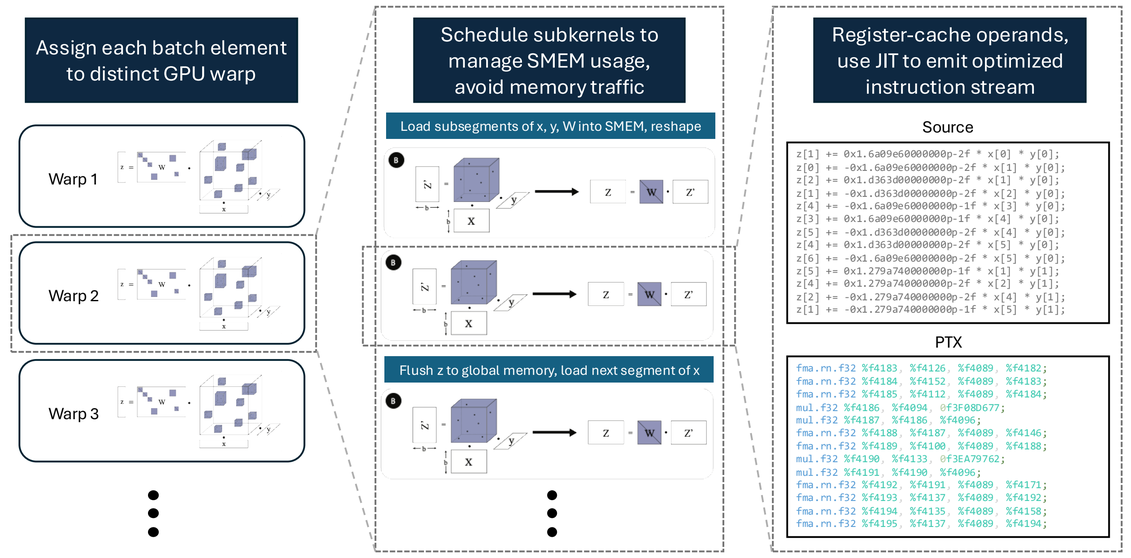
We introduce an *OpenEquivariance*, an open-source CUDA kernel generator for the Clebsch-Gordon tensor product. **5-6x end-to-end lower runtime** to train / perform inference on chemical foundation models like Nequip, MACE.

Key Features:

- 10x performance boost over e3nn to compute CG tensor product and 1st, 2nd derivatives.
- On-par performance with closed-source NVIDIA cuEquivariance v0.4.0.
- e3nn-compatible interface supports both AMD + NVIDIA.



Kernel Design Strategies



A sample of our contributions (see paper for more details):

- Tensor products are dispatched to warps that operate asynchronously.
- Warps manage a partition of GPU SRAM by breaking operation into a series of subkernels (one per nonzero sparse block), partially staging arguments.
- Subkernels unroll loops, exploit tensor structure to achieve high performance.

Performance Benchmarks

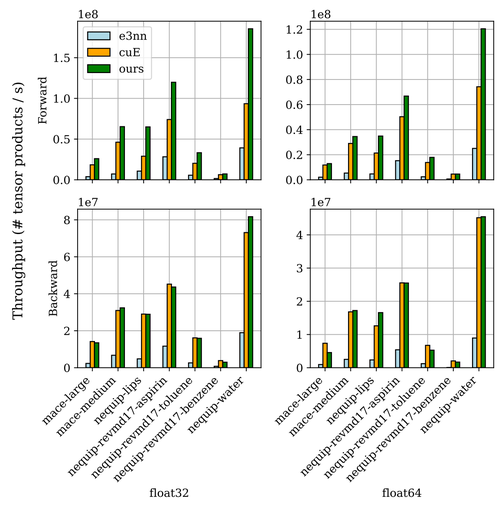


Figure 1. Unfused Nequip / MACE tensor product throughput, A100, batch 50K.

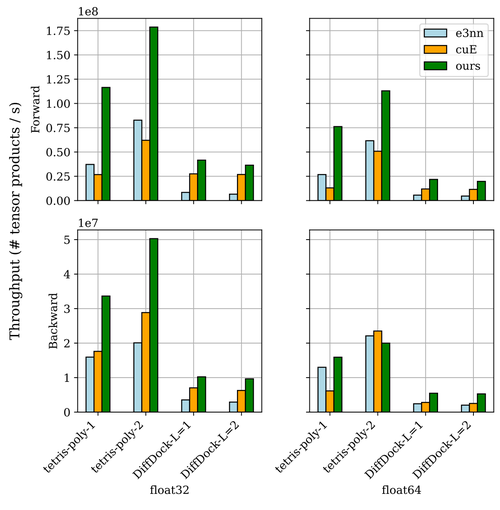


Figure 2. Unfused DiffDock + Tetris tensor product throughput, A100, batch 50K.

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

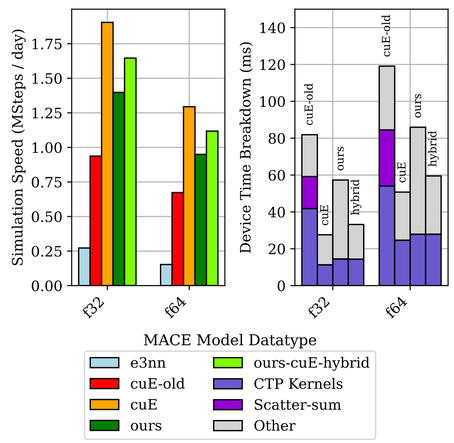


Figure 3. MACE A100 inference speed.

Acknowledgements and Paper Link

V. Bharadwaj was supported by a Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0022158.

Work to appear at ACDA25 - scan the code to read the paper!