The Price of Freedom: Exploring Tradeoffs between Expressivity and Computational Efficiency in Equivariant Tensor Products

YuQing Xie [1] Ameya Daigavane [1] Mit Kotak [1] Tess Smidt [1]

Abstract

$E(3)$-equivariant neural networks have recently demonstrated success across a wide range of 3D modelling tasks. A fundamental operation in these networks is the tensor product that interacts two geometric features in an equivariant manner to create new features. Due to the high computational complexity of the tensor product, significant effort has been invested to optimize the runtime of this operation. For example, Luo et al. (2024) proposed the Gaunt tensor product, promising a significant speedup over the naive implementation of the tensor product. Here, we perform a careful systematic analysis of the runtimes and expressivity of different tensor product implementations. We applied this analysis to Clebsch-Gordan tensor product (CGTP), Gaunt tensor product (GTP), and fused tensor product (FTP). We find that the naive implementation of CGTP can be improved by leveraging sparsity of the Clebsch-Gordan coefficients. Further, we show that the original implementation proposed in Luo et al. (2024) using 2D Fourier basis can be improved by projecting to the sphere S^2 instead which we call grid GTP. In addition, we show that the improvements of GTP and FTP come at a *cost of expressivity* compared to CGTP. In fact, in some settings they are *asymptotically slower* than the sparse version of CGTP. Finally, we provide some experimental benchmarks for CGTP and GTP. Our code is available here.

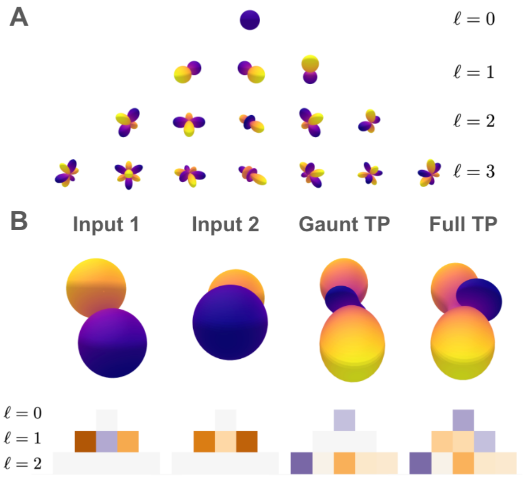

Figure 1. (A) Spherical harmonics are basis functions that transform as specific irreps of $O(3)$. (B) Gaunt Tensor Products accelerate tensor products using projections on the sphere, however this eliminates certain paths such as antisymmetric outputs.

1. Introduction

Incorporating symmetries when modelling complex systems enables efficient and robust learning. This phenomenon has been aptly demonstrated by $E(3)$-equivariant models(Park et al., 2020; Thomas et al., 2018; Weiler et al., 2018; Kondor, 2018; Kondor et al., 2018) which faithfully model the symmetries of 3D systems. This has resulted in strong performance across a wide range of scientific problems – for example, in molecular force fields (Batzner et al., 2022; Musaelian et al., 2023; Batatia et al., 2022), catalyst discovery (Liao & Smidt, 2023), generative models (Hoogeboom et al., 2022), charge density prediction (Fu et al., 2024), and protein structure prediction (Lee et al., 2022; Jumper et al., 2021).

The group $E(3)$ consisting of all rotations, translations and reflections in 3 dimensions; we say a model is $E(3)$-equivariant if it satisfies:

$$f(g \cdot x) = g \cdot f(x) \quad \forall g \in E(3), x \in X \tag{1}$$

$E(3)$-equivariant neural networks work with features that transform as irreducible representations of $SO(3)$, termed 'irreps', as described in Section 2. These irreps can be described by a number L, indicating how they transform under $E(3)$. To interact these irreps, a special 'tensor product'

[1] Massachusetts Institute of Technology. Correspondence to: YuQing Xie <xyuqing@mit.edu>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024.

operation is performed, similar to how features are added and multiplied with each other in a neural network. As described in Section 3, the well-studied Clebsch-Gordan (Varshalovich et al., 1988) coefficients can be used to define a tensor product. The Clebsch-Gordan Tensor Product has a time complexity[1] of $\mathcal{O}(L^5)$ as we show in Appendix B, which can quickly become expensive for larger L. This scaling has limited the direct application of $E(3)$-equivariant neural networks to larger systems; and there is now much interest in optimizing several key operations within these neural networks.

One such optimization was identified by (Passaro & Zitnick, 2023) for the special case of when one of the inputs is derived from the spherical harmonics. Under a suitable rotation, the Clebsch-Gordan coefficients become sparse, allowing for a runtime of $\mathcal{O}(L^3)$.

To optimize the general case, Luo et al. (2024) proposed the Gaunt Tensor Product which claims to bring down the complexity of the tensor product to $\mathcal{O}(L^3)$. While this represents exciting progress, we have identified two limitations of the Gaunt Tensor Product:

- We show that the output spaces of the Gaunt Tensor Product and the Clebsch-Gordan Tensor Product are not directly comparable. In particular, when normalizing for this difference, the Clebsch-Gordan Tensor Product is *asymptotically faster* than the Gaunt Tensor Product in some settings.
- The Gaunt Tensor Product is restricted to what it can output. In particular, it cannot represent *antisymmetric* interactions, which can be important in many physical systems. Please see Section 4.1 for an example.

Furthermore, the implementation of the Gaunt Tensor Product as originally proposed can be simplified without compromising on computational complexity. Instead of using a 2D Fourier basis as implemented in Luo et al. (2024), we can directly project onto the sphere S^2. In fact, using a special recurrence relation for the spherical harmonics, this idea enables an asymptotically faster implementation of the Gaunt tensor product in $\mathcal{O}(L^2 \log^2 L)$ time.

Next, we provide some background for understanding the operations of $E(3)$-equivariant neural networks. In Section 3, we discuss the asymptotics of various tensor products, including the FusedTensorProduct operation introduced by Unke & Maennel (2024). In Appendix B, we show the true runtimes of these tensor products on actual CPU and GPU hardware.

2. Irreducible Representations of $E(3)$

A representation ρ of a group G maps each group element g to a bijective linear transformation $\rho(g) \in \mathrm{GL}(V)$, where V is some vector space. Representations must preserve the group multiplication property:

$$\rho(g \cdot h) = \rho(g) \circ \rho(h) \quad \forall g, h \in G \tag{2}$$

Thus, the representation ρ defines a group action on a vector space V. The dimension of the representation ρ is simply defined as the dimension of the vector space V.

There may be subspaces $W \subset V$ which are left invariant under actions of $\rho(g)$ for all $g \in G$. If this is the case, then restricting to W also gives a representation $\rho|_W(g) \in \mathrm{GL}(W)$. If there is no nontrivial W, then we say the representation ρ is an irreducible representation (irrep).

To build $E(3)$-equivariant neural networks, the irreducible representations of $E(3)$ play a key role. Because $E(3)$ is not a compact group, the usual approach has been to consider irreducible representations of the group $SO(3)$ of 3D rotations, and compose them with the representation in which translations act as the identity:

$$\rho(R, T) = \rho'(R) \tag{3}$$

This is why translations are often handled in $E(3)$-equivariant neural networks by centering the system or only using relative vectors.

The 'scalar' representation ρ_{scalar} representation of $SO(3)$ is defined as:

$$\rho_{\text{scalar}}(R) = \mathrm{id} \quad \forall R \in SO(3) \tag{4}$$

and is of dimension 1 over $V = \mathbb{R}$. Elements of $\mathbb{R}$ are unchanged by any rotation R. We call such elements 'scalars' to indicate that they transform under the 'scalar' representation of $SO(3)$. An example of a 'scalar' element could be mass of an object, which does not change under rotation of coordinate frames.

Let $T(R) \in \mathbb{R}^{3\times 3}$ be the rotation matrix corresponding to a rotation $R \in SO(3)$. Then, the 'vector' representation of $SO(3)$ is defined as:

$$\rho_{\text{vector}}(R) = T(R) \quad \forall R \in SO(3) \tag{5}$$

and is of dimension 3 over $V = \mathbb{R}^3$. The name arises from the way vectors in $\mathbb{R}^3$ transform under a rotation of the coordinate frame. We call such elements 'vectors' to indicate that they transform under the 'vector' representation of $SO(3)$. For example, the velocity and position of an object in a certain coordinate frame are 'vectors'.

Weyl's theorem for the Lie group $SO(3)$ states that all finite-dimensional representations of $SO(3)$ are equivalent to direct sums of irreducible representations. The irreducible

[1] Note that (Passaro & Zitnick, 2023) claims a runtime of $\mathcal{O}(L^6)$ for this tensor product. In Appendix B, we show that this runtime is actually $\mathcal{O}(L^4)$.

representations of $SO(3)$ are indexed by an integer $\ell \geq 0$, with dimension $2\ell + 1$. $\ell = 0$ corresponds to the 'scalar' representation, while $\ell = 1$ corresponds to the 'vector' representation above. We will often use m, where $-\ell \leq m \leq \ell$, to index of each of the $2\ell + 1$ components.

We say that a quantity $\mathbf{x} \in \mathbb{R}^{2\ell+1}$ is a ℓ irrep, if it transforms as the irreducible representation ('irrep') of $SO(3)$ indexed by ℓ. If $\mathbf{x}_1$ is a ℓ_1 irrep and $\mathbf{x}_2$ is an ℓ_2 irrep, we say that $(\mathbf{x}_1, \mathbf{x}_2)$ is a direct sum of ℓ_1 and ℓ_2 irreps, which we call a (ℓ_1, ℓ_2) 'rep'. Weyl's theorem states that all reps are a direct sum of ℓ_i irreps, possibly with repeats over ℓ_i: $\mathbf{x} = \oplus_{\ell_i} \mathbf{x}^{(\ell_i)}$. The multiplicity of an irrep in a rep is exactly the number of repeats.

3. Tensor Products

Given two reps $\mathbf{x}$ and $\mathbf{y}$, how can we construct new reps by interacting $\mathbf{x}$ and $\mathbf{y}$? A very general interaction is the tensor product, which is essentially an outer product. For two rank 1 tensors $\mathbf{x}, \mathbf{y}$, the tensor product gives a rank 2 tensor (matrix) $\mathbf{Z}_{ij} = (\mathbf{x} \otimes \mathbf{y})_{ij} = \mathbf{x}_i \mathbf{y}_j$. We call the tensor product of two irreps a tensor product rep.

3.1. Clebsch-Gordan Tensor Product

Suppose we have 2 reps both of which have been separated into a sum of irreps. The tensor product of these reps can be written as

$$\mathbf{x} \otimes \mathbf{y} = \oplus_{\substack{\mathbf{x}^{(\ell_1)} \in \mathbf{x} \\ \mathbf{y}^{(\ell_2)} \in \mathbf{y}}} (\mathbf{x}^{(\ell_1)} \otimes \mathbf{y}^{(\ell_2)}) \tag{6}$$

a new rep which is the sum of tensor product reps.

We can reduce the tensor product reps back into a direct sum of irreps using Clebsch-Gordan coefficients, giving us

$$\mathbf{x}^{(\ell_1)} \otimes \mathbf{y}^{(\ell_2)} = \bigoplus_{\ell_3} (\mathbf{x}^{(\ell_1)} \otimes_{\text{CG}} \mathbf{y}^{(\ell_2)})^{(\ell_3)} \tag{7}$$

where

$$\begin{aligned} &(\mathbf{x}^{(\ell_1)} \otimes_{\text{CG}} \mathbf{y}^{(\ell_2)})^{(\ell_3)}_{m_3} \\ &= \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C^{(\ell_3, m_3)}_{\ell_1, m_1, \ell_2, m_2} \mathbf{x}^{(\ell_1)}_{m_1} \mathbf{y}^{(\ell_2)}_{m_2}. \end{aligned} \tag{8}$$

Therefore the original tensor product can also be rewritten as a direct sum of irreps. This defines the Clebsch-Gordan tensor product (CGTP)

$$\mathbf{x} \otimes_{\text{CG}} \mathbf{y} = \oplus_{\substack{\mathbf{x}^{(\ell_1)} \in \mathbf{x} \\ \mathbf{y}^{(\ell_2)} \in \mathbf{y}}} (\mathbf{x}^{(\ell_1)} \otimes_{\text{CG}} \mathbf{y}^{(\ell_2)}). \tag{9}$$

Note that because we simply change the basis in full CGTP from a tensor product rep to a sum of irreps, we **do not lose any information**.

The decomposition of tensor product reps into irreps is well studied. There are selection rules, which tell us the only ℓ_3 in Equation 8 that can be non-zero satisfy $|\ell_1 - \ell_2| \leq \ell_3 \leq \ell_1 + \ell_2$ (Varshalovich et al., 1988).

3.2. Gaunt Tensor Product

The Gaunt tensor product (GTP) as introduced by Luo et al. (2024) uses the intimate connection between spherical harmonics, irreps, and spherical signals. We define this more precisely in Appendix A. Any rep of form $(0, 1, \ldots, L)$ can be interpreted as coefficients for the spherical harmonics and hence corresponds to a function on S^2.

In particular, given two $(0, 1, \ldots, L)$ reps $\mathbf{x}$ and $\mathbf{y}$, let $f_{\mathbf{x}} = \text{ToSphere}(\mathbf{x})$ and $f_{\mathbf{y}} = \text{ToSphere}(\mathbf{y})$ be the associated functions on S^2. Taking the pointwise product of $f_{\mathbf{x}}$ and $f_{\mathbf{y}}$ on S^2 gives us a new function $f_{\mathbf{x}} \cdot f_{\mathbf{y}}$, also on S^2. Then, converting back to irreps gives us the Gaunt tensor product:

$$\mathbf{x} \otimes_{\text{GTP}} \mathbf{y} = \text{FromSphere}(f_{\mathbf{x}} \cdot f_{\mathbf{y}}) \tag{10}$$

From the same selection rules as the CGTP, we will only have irreps up to $2L$. However, spherical signals only have one copy of each irrep. So $\mathbf{x} \otimes_{\text{GTP}} \mathbf{y}$ will be a $(0, 1, \ldots, 2L)$ rep. This already highlights one key difference already between the GTP and the CGTP: the multiplicities of each irrep in the GTP can be **at most one**. However, Equation 8 shows that the multiplicities of some irreps in the CGTP can be $\Theta(L^2)$. What happens is that output irreps of the same ℓ get weighted and summed. Hence, compared to the CGTP, **GTP loses information**. Even adding additional per-irrep learned weights does not help the expressivity of the GTP, as we prove in Appendix C.

This distinction is exactly where the speedup from the Gaunt tensor product arises: the outputs of both operations are not directly comparable with each other. To address this discrepancy, we consider 3 different settings to analyse the theoretical and empirical runtimes of different tensor products:

- **Single Input, Single Output (SISO)**:
$$L \otimes L \to L$$
- **Single Input, Multiple Output (SIMO)**:
$$L \otimes L \to (0, 1, \ldots, 2L)$$
- **Multiple Input, Multiple Output (MIMO)**:
$$(0, 1, \ldots, L) \otimes (0, 1, \ldots, L) \to (0, 1, \ldots, 2L)$$

where we assume $\ell_1 = \mathcal{O}(L), \ell_2 = \mathcal{O}(L)$. We show that the choice of 'most efficient' tensor product depends on the

Table 1. Asymptotic runtimes of various tensor products for different output settings. The best performing tensor products for each output settings are highlighted in green. In the MIMO setting, the Clebsch-Gordan tensor products are highlighted in red to indicate that they can output irreps with multiplicity > 1 , unlike the Gaunt tensor products.

Tensor Product	SISO	SIMO	MIMO
Clebsch-Gordan (Naive)	$\mathcal{O}(L^3)$	$\mathcal{O}(L^4)$	$\mathcal{O}(L^6)$
Clebsch-Gordan (Sparse)	$\mathcal{O}(L^2)$	$\mathcal{O}(L^3)$	$\mathcal{O}(L^5)$
Gaunt (Original)	$\mathcal{O}(L^2 \log L)$	$\mathcal{O}(L^3)$	$\mathcal{O}(L^3)$
Gaunt (Naive Grid)	$\mathcal{O}(L^2 \log L)$	$\mathcal{O}(L^3)$	$\mathcal{O}(L^3)$
Gaunt (S2FFT Grid)	$\mathcal{O}(L^2 \log L)$	$\mathcal{O}(L^2 \log^2 L)$	$\mathcal{O}(L^2 \log^2 L)$
Fused Tensor Product	$\mathcal{O}(L^3)$	$\mathcal{O}(L^3)$	$\mathcal{O}(L^3)$

setting. Table 1 summarizes the first of our contributions: a tight analysis of the runtimes of different tensor products. We defer details to Appendix B.

In addition, note that GTP is a symmetric operation. Hence, antisymmetric outputs cannot appear. For example, GTP cannot be used to compute cross products, because $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$. In Section 4.1, we show that this implies that the Gaunt tensor product is incapable of solving a simple task of classifying chiral 3D structures.

3.3. Fused Tensor Product

The new `e3x` framework introduces another interaction which they called `FusedTensor` (Unke & Maennel, 2024). The main motivation for this interaction is that a tensor product rep is a matrix and we can interact 2 tensor product reps through matrix multiplication.

Hence, FTP first takes each input and embeds the irreps in a single large enough tensor product rep using Clebsch-Gordan coefficients. After doing so, we can matrix multiply the tensor product reps. Finally, we can decompose the resulting tensor product rep back into irreps. Details are provided in Section B.3.

Similar to GTP, FTP only outputs one copy of each possible output irrep. Hence, the output irreps of the same irreps get weighted and summed together and **FTP loses information** in the same way as GTP. However in contrast to GTP, FTP is not a symmetric operation so we can have antisymmetric tensor product terms.

4. Experiments

4.1. An Example with Antisymmetry: Classifying 3D Tetris Pieces

We consider a simple task of classifying 8 different 3D Tetris-like pieces, shown in Figure 2.

Note that the first two pieces are non-superimposable mirror reflections of each other; they are *chiral*.

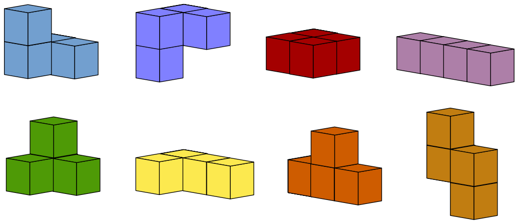

Figure 2. The 8 different 3D Tetris pieces, with the first two pieces being mirror images of each other.

We use a simple message-passing neural network, described in Appendix D, using either the Gaunt and Clebsch-Gordan tensor products. Our network architecture is almost identical to that of NequIP (Batzner et al., 2022).

Given a randomly oriented 3D structure, the network needs to predict which of the 8 tetris pieces it corresponds to.

The pieces are normalized such that the side length of each cube is 1. When represented as a graph, the center of each cube is a node. We instantiate the network with $d_{\text{max}} = 1.1$ so that the centers are connected only to its immediately adjacent centers. The networks finally outputs $\mathbf{x} = 7 \times 0e + 1 \times 0o$ irreps. (As a reminder, $0e$ are scalars and $0o$ are pseudoscalars). The logits and predicted probabilities are then computed by:

$$\begin{aligned} l_0 &= \mathbf{x}^{(0o)} \times \mathbf{x}^{(0e)_0} \\ l_1 &= -\mathbf{x}^{(0o)} \times \mathbf{x}^{(0e)_0} \\ l_i &= \mathbf{x}^{(0e)_i} \quad \text{for } i \geq 2 \\ p_i &= \text{softmax}(l_i) \end{aligned} \tag{11}$$

It is clear that defining the logits in this manner preserves the rotational and reflection symmetries. The predictions are clearly invariant under rotations (as they are $\ell = 0$ irreps), and under reflections: $\mathbf{x}^{(0o)} \to -\mathbf{x}^{(0o)}$ but $\mathbf{x}^{(0e)_i} \to \mathbf{x}^{(0e)_i}$.

We set the number of message-passing steps T to be 3, to allow the interactions $1o \otimes 1o \to 1e$ and then $1e \otimes 1o \to 0o$, so that the pseudoscalar can be created. The degree of spherical harmonics is kept as $\ell = 4$. The irreps of the hidden layers are restricted to some cutoff L, which is varied from 1 to 4 to vary the expressivity of the network. The network is trained with the Adam optimizer with learning rate 0.01 to minimize the standard cross-entropy loss to one-hot encoded labels for each of the 8 pieces.

As shown in Figure 3, the network is very easily able to solve this task with the Clebsch-Gordan tensor product, but the same network parametrized with the Gaunt tensor product is unable to distinguish between the two chiral pieces. Adding more channels or incorporating the pseudo-spherical harmonics (which have the opposite parity of the spherical harmonics under reflection) did not help. The fundamental

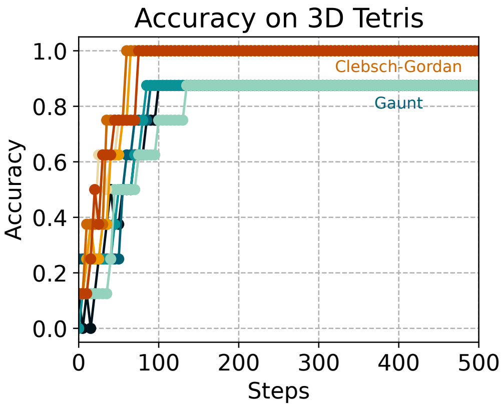

Figure 3. Training curves of networks trained with different tensor products on the 3D Tetris task. The maximum L is varied from 1 to 4. All of the Clebsch-Gordan networks attain 100% accuracy while none of the Gaunt networks do.

failure is the inability to create the $1e$ term via $1o \otimes 1o \to 1e$ because this is the cross product, ie an antisymmetric operation. Indeed, there is no way to create a pseudoscalar using the Gaunt tensor product in this setting.

4.2. Benchmarking Different Tensor Products

We report wall clock times, peak throughput (GFLOP/s), and peak bandwidth (GB/s) for Clebsch-Gordan (Naive), Clebsch-Gordan (Sparse), Gaunt (Original) and Gaunt (Naive Grid) Table 1.

We analyze results for the MIMO setting on a NVIDIA RTX A5500 GPU in Figure 4. Results on the CPU and for other tensor products can be found in Appendix E.

- The faster wall-times for GTP (Naive Grid, Original Grid) can be attributed to high throughput and bandwidth. On the other hand, CGTP (Sparse) is bottlenecked by low throughput and bandwidth utilization.

- CGTP (Sparse) is slower than CGTP (Naive) despite having lower computational complexity. This is due to irregular memory access patterns leading to poor spatial and temporal locality.

- Unlike matrix multiplication, the throughput efficiency (NVIDIA, 2024) for the CGTP (Sparse, Original) product operations doesn't improve with increasing L_{max}. Our current benchmarks do not do any batching, possibly leading to low GPU utilization. In the future, we plan to extend the benchmarks to account for batch dimensions.

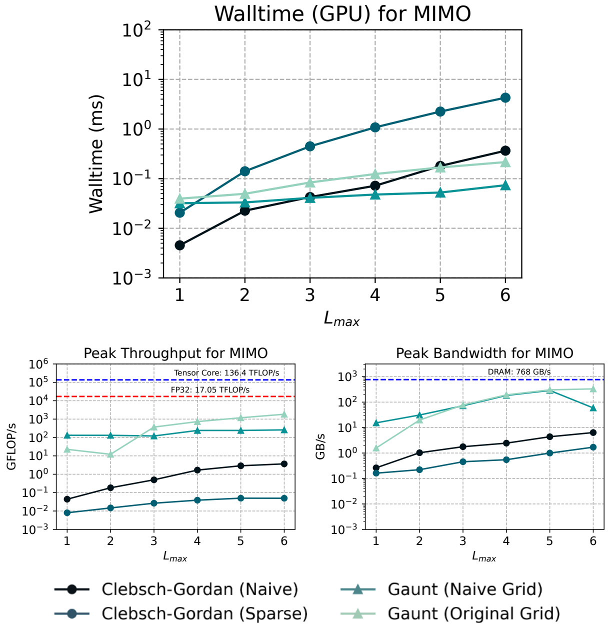

Figure 4. Analysis of MIMO performance for different tensor products on RTX A5500: Walltime (top), Peak GFLOP/s (middle), Peak GB/s (bottom)

5. Conclusion

We analyzed the asymptotic runtimes of various tensor product operations and show most of the improved performance comes at the price of expressivity. We further benchmarked CGTP and GTP implementations to see their performance in practice. Despite better asymptotics, we find Clebsch-Gordan (Sparse) to perform worse due to low throughput and bandwidth utilization, caused by irregular memory access patterns.

References

Batatia, I., Kovacs, D. P., Simm, G. N. C., Ortner, C., and Csanyi, G. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=YPpSngE-ZU.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. 13, May 2022. URL https://doi.org/10.1038/s41467-022-29939-5.

Beentjes, C. H. Quadrature on a spherical surface. *Working* *note available on the website http://people. maths. ox. ac. uk/beentjes/Essays*, 2015.

Cobb, O., Wallis, C. G., Mavor-Parker, A. N., Marignier, A., Price, M. A., d'Avezac, M., and McEwen, J. Efficient generalized spherical cnns. In *International Conference on Learning Representations*.

Fu, X., Rosen, A., Bystrom, K., Wang, R., Musaelian, A., Kozinsky, B., Smidt, T., and Jaakkola, T. A recipe for charge density prediction, 2024.

Healy, D. M., Rockmore, D. N., Kostelec, P. J., and Moore, S. FFTs for the 2-sphere-improvements and variations. *Journal of Fourier analysis and applications*, 9:341–385, 2003.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d, 2022.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kondor, R. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials, 2018. URL https://arxiv.org/abs/1803.01588.

Kondor, R., Lin, Z., and Trivedi, S. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network, 2018. URL https://arxiv.org/abs/1806.09231.

Lebedev, V. I. Quadratures on a sphere. *USSR Computational Mathematics and Mathematical Physics*, 16(2): 10–24, 1976.

Lee, J. H., Yadollahpour, P., Watkins, A., Frey, N. C., Leaver-Fay, A., Ra, S., Cho, K., Gligorijevic, V., Regev, A., and Bonneau, R. Equifold: Protein structure prediction with a novel coarse-grained structure representation. *bioRxiv*, 2022. doi: 10.1101/2022.10.07.511322. URL https://www.biorxiv.org/content/early/2022/10/08/2022.10.07.511322.

Liao, Y.-L. and Smidt, T. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KwmPfARgOTD.

Luo, S., Chen, T., and Krishnapriyan, A. S. Enabling efficient equivariant operations in the fourier basis via gaunt tensor products. *arXiv preprint arXiv:2401.10216*, 2024.

McLaren, A. D. Optimal numerical integration on a sphere. *Mathematics of Computation*, 17(84):361–383, 1963.

Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C. J., Kornbluth, M., and Kozinsky, B. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, 2023.

NVIDIA. Matrix multiplication background user's guide. Technical report, NVIDIA, 2024. URL https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html. Accessed on July 13, 2024.

Park, C. W., Kornbluth, M., Vandermause, J., Wolverton, C., Kozinsky, B., and Mailoa, J. P. Accurate and scalable multi-element graph neural network force field and molecular dynamics with direct force architecture, 2020. URL https://arxiv.org/abs/2007.14444.

Passaro, S. and Zitnick, C. L. Reducing so (3) convolutions to so (2) for efficient equivariant gnns. In *International Conference on Machine Learning*, pp. 27420–27438. PMLR, 2023.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018. URL https://arxiv.org/abs/1802.08219.

Unke, O. T. and Maennel, H. E3x: $\mathrm{E}(3)$-equivariant deep learning made easy. *arXiv preprint arXiv:2401.07595*, 2024.

Varshalovich, D. A., Moskalev, A. N., and Khersonskii, V. K. *Quantum theory of angular momentum*. World Scientific, 1988.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018. URL https://arxiv.org/abs/1807.02547.

Xin, H., Zhou, Z., An, D., Yan, L.-Q., Xu, K., Hu, S.-M., and Yau, S.-T. Fast and accurate spherical harmonics products. *ACM Trans. Graph.*, 40(6):280–1, 2021.

Yang, C. Hierarchical roofline analysis: How to collect data using performance tools on intel cpus and nvidia gpus, 2020. URL https://arxiv.org/abs/2009.02449.

A. Spherical Harmonics

The spherical harmonics are intimately connected to the representations of $SO(3)$ and play a key role in the Gaunt tensor product.

We define the spherical coordinates (r, θ, φ) as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\sin\theta\cos\varphi \\ r\sin\theta\sin\varphi \\ r\cos\theta \end{bmatrix} \tag{12}$$

for $\theta \in [0, \pi), \varphi \in [0, 2\pi)$.

The spherical harmonics $Y_{\ell,m}$ are a set of functions $S^2 \to \mathbb{R}$ indexed by (ℓ, m), where again $\ell \geq 0, -\ell \leq m \leq \ell$. Here, $S^2 = \{(r, \theta, \phi) \mid r = 1\}$ denotes the unit sphere.

Indeed, as suggested by the notation, the spherical harmonics are closely related to the irreducible representations of $SO(3)$. Let Y_ℓ be the concatenation of all $Y_{\ell,m}$ over all m for a given ℓ:

$$Y_\ell(\theta, \phi) = \begin{bmatrix} Y_{\ell,-\ell}(\theta, \phi) \\ Y_{\ell,-\ell+1}(\theta, \phi) \\ \dots \\ Y_{\ell,\ell}(\theta, \phi) \end{bmatrix} \tag{13}$$

When we transform the inputs to $Y_\ell(\theta, \phi)$, the output transforms as a ℓ irrep.

The spherical harmonics satisfy orthogonality conditions:

$$\int_{S^2} Y_{\ell_1,m_1} \cdot Y_{\ell_2,m_2} \, dS^2 = \delta_{\ell_1\ell_2}\delta_{m_1m_2} \tag{14}$$

where:

$$\int_{S^2} f \cdot g \, dS^2 = \int_{\theta=0}^{\pi} \int_{\varphi=0}^{2\pi} f(\theta, \varphi) g(\theta, \varphi) \sin\theta d\theta d\varphi \tag{15}$$

The orthogonality property allows us to treat the spherical harmonics as a basis for functions on S^2. We can linearly combine the spherical harmonics using irreps to approximate arbitrary functions on the sphere. Given a $(0, 1, \dots, L)$ rep $\mathbf{x} = (\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})$, we can associate the function $f_\mathbf{x} : S^2 \to \mathbb{R}$ as:

$$f_\mathbf{x}(\theta, \varphi) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \mathbf{x}_m^{(\ell)} Y_{\ell,m}(\theta, \varphi) \tag{16}$$

The function $f_\mathbf{x}$ is uniquely determined by $\mathbf{x}$. In particular, by the orthogonality of the spherical harmonics (Equation 14), we can recover the $\mathbf{x}_m^{(\ell)}$ component:

$$\mathbf{x}_m^{(\ell)} = \int_{S^2} f_\mathbf{x} \cdot Y_{\ell,m} \, dS^2 \tag{17}$$

Thus, we can define the operations $\mathrm{ToSphere}$ and $\mathrm{FromSphere}$:

$$\mathbf{x} \xrightarrow{\mathrm{ToSphere}} f_\mathbf{x} \xrightarrow{\mathrm{FromSphere}} \mathbf{x} \tag{18}$$

B. Runtime Analysis

Here, we provide further details about the asymptotic analysis of runtimes for different tensor products.

B.1. Clebsch-Gordan Tensor Product

The tensor product operation is defined as:

$$\otimes_{\mathrm{CG}}\mathbf{x}^{(\ell_2)})^{(\ell_3)}_{m_3} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C^{(\ell_3,m_3)}_{\ell_1,m_1,\ell_2,m_2}\mathbf{x}^{(\ell_1)}_{m_1}\mathbf{x}^{(\ell_2)}_{m_2} \tag{19}$$

where C denotes the Clebsch-Gordan (CG) coefficients which can be precomputed.

B.1.1. Naive runtime

Let $L = \max(\ell_1, \ell_2, \ell_3)$. From Equation 19, for each m_3, we would need to sum over m_1, m_2 which range from $-\ell_1$ to ℓ_1 and $-\ell_2$ to ℓ_2 respectively. Hence, we expect $\mathcal{O}(L^2)$ operations. To compute the values for all m which range from $-\ell_3$ to ℓ_3, we see that computing a single $\ell_1 \otimes \ell_2 \to \ell_3$ tensor product requires $\mathcal{O}(L^3)$ operations.

B.1.2. Optimized runtime with sparsity

However, the CG coefficients are sparse. In the complex basis for the irreps, $C^{(\ell_3,m)}_{\ell_1,m_1,\ell_2,m_2}$ is nonzero only if $m_1 + m_2 = m_3$. Transforming to the real basis for the irreps, this condition becomes $\pm m_1 \pm m_2 = m_3$. In either case for a fixed m_1 and m, we only ever need to sum over a constant number of m_2's rather than $\mathcal{O}(L)$ of them as naively expected. Therefore an implementation taking this sparsity into account gives us a runtime of $\mathcal{O}(L^2)$. This optimization was noted in Cobb et al..

B.2. Gaunt Tensor Product

The Gaunt Tensor Product (GTP) is based on the decomposition of a product of spherical harmonic functions back into spherical harmonics (Luo et al., 2024). In particular, suppose one of our inputs $\mathbf{x}^{(\ell_1)}$ transforms as a direct sum of irreps up to some cutoff L (ie. ℓ_1 ranges from $0, \dots, L$). We can view these irreps as coefficients of spherical harmonics which gives a spherical signal $F_1(\theta, \varphi) = \sum_{\ell_1,m_1} \mathbf{x}^{(\ell_1)}_{m_1} Y_{\ell_1,m_1}(\theta,\varphi)$. We similarly construct $F_2(\theta,\varphi) = \sum_{\ell_2,m_2} \mathbf{x}^{(\ell_2)}_{m_2} Y_{\ell_2,m_2}(\theta,\varphi)$.

Taking the product of these spherical signals gives a new signal $F_3(\theta,\varphi) = F_1(\theta,\varphi)F_2(\theta,\varphi)$. This new signal can be decomposed into spherical harmonics which we use to define the GTP. This results in

$$F_3(\theta,\varphi) = \sum_{\ell_3,m_3} (\mathbf{x}^{(\ell_1)} \otimes_{\mathrm{GTP}} \mathbf{x}^{(\ell_2)})^{(\ell_3)}_{m_3} Y_{\ell_3,m_3}(\theta,\varphi). \tag{20}$$

B.2.1. 2D Fourier basis

Luo et al. (2024) describe an implementation which decomposes spherical harmonics into a 2D Fourier basis in their original paper introducing GTP. This also turns out to be the same implementation in Xin et al. (2021). We describe their procedure here.

Note that for any $\ell \leq L$ we can always write the spherical harmonics in the 2D Fourier basis:

$$Y_{\ell,m}(\theta,\varphi) = \sum_{-L \leq u,v \leq L} y^{\ell,m}_{u,v} e^{i(u\theta + v\varphi)} \tag{21}$$

for some coefficients $y^{\ell,m}_{u,v}$.

Hence, any signal $\mathbf{x}^{(\ell)}_m$ can be encoded as

$$F_1(\theta,\varphi) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \sum_{-L \leq u,v \leq L} \mathbf{x}^{(\ell)}_m y^{\ell,m}_{u,v} e^{i(u\theta+v\varphi)} = \sum_{-L \leq u,v \leq L} \left(\sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \mathbf{x}^{(\ell)}_m y^{\ell,m}_{u,v} \right) e^{i(u\theta+v\varphi)}. \tag{22}$$

We identify the encoding

$$\mathbf{x}_{u,v} = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \mathbf{x}^{(\ell)}_m y^{\ell,m}_{u,v}. \tag{23}$$

One can observe that the $y_{u,v}^{\ell,m}$ are sparse and only nonzero when $m = \pm v$. Therefore, finding $\mathbf{x}_{u,v}$ if we have a set of irreps is $\mathcal{O}(L)$ and it is $O(1)$ if we only want one irrep. Because there are $\mathcal{O}(L^2)$ possible values for u, v, encoding into the 2D Fourier is $\mathcal{O}(L^3)$ if we encode all irreps up to L or $\mathcal{O}(L^2)$ if encoding a single irrep.

For 2 functions of θ, φ encoded using a 2D Fourier basis $\mathbf{x}^1_{u,v}, \mathbf{x}^2_{u,v}$, we can compute their product using a standard 2D FFT in $\mathcal{O}(L^2 \log L)$ time. This gives some output encoded as $\mathbf{y}_{u,v}$ where now u, v range from $-2L, \dots, 2L$ to capture all information.

Finally, we decode the resulting function in the 2D Fourier basis back into a spherical harmonic basis to extract the output irreps. Suppose $-L \le u, v \le L$. We can always write

$$e^{i(u\theta+v\varphi)} = F^{\perp}_{u,v}(\theta,\varphi) + \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell} z_{u,v}^{\ell,m} Y_{\ell,m}(\theta,\varphi) \tag{24}$$

where $F^{\perp}_{u,v}(\theta,\varphi)$ is some function in the space orthogonal to that spanned by the spherical harmonics. By construction, our output signal is always in the space spanned by the spherical harmonics so the orthogonal parts cancel. Hence we can write

$$\sum_{-2L\le u,v\le 2L} \mathbf{y}_{u,v} e^{i(u\theta+v\varphi)} = \sum_{-2L\le u,v\le 2L} \mathbf{y}_{u,v} \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell} z_{u,v}^{\ell,m} Y_{\ell,m}(\theta,\varphi) \tag{25}$$

$$= \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell} \left(\sum_{-2L\le u,v\le 2L} \mathbf{y}_{u,v} z_{u,v}^{\ell,m} \right) Y_{\ell,m}(\theta,\varphi) \tag{26}$$

Hence we identify:

$$\mathbf{y}^{\ell}_{m} = \sum_{-2L\le u,v\le 2L} \mathbf{y}_{u,v} z_{u,v}^{\ell,m}. \tag{27}$$

Once again, we can note that $z_{u,v}^{\ell,m}$ must be sparse and is only nonzero when $v = \pm m$. Hence, evaluating the above takes $\mathcal{O}(L)$ time since we sum over $\mathcal{O}(L)$ values of u paired with constant number of v's. If we only extract one irrep, then we range over $\mathcal{O}(L)$ values of m giving $\mathcal{O}(L^2)$ runtime. If we extract all irreps up to $2L$ this becomes $\mathcal{O}(L^3)$.

B.2.2. Grid tensor product

Rather than use a 2D Fourier basis, we can instead represent the signal by directly giving its value for a set of points on the sphere. Quadrature on the sphere is a well-studied topic (Beentjes, 2015; Lebedev, 1976); in general, $\mathcal{O}(L^2)$ points are needed to exactly integrate spherical harmonics upto degree L (McLaren, 1963). For this section, consider a product grid on the sphere formed by the Cartesian product of two 1D grids for θ and φ with $\mathcal{O}(L)$ points each, for a total of $\mathcal{O}(L^2)$ points.

We can write:

$$F_1(\theta_j,\varphi_k) = \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell} \mathbf{x}^{(\ell)}_m Y_{\ell,m}(\theta_j,\varphi_k) = \sum_{\ell=0}^{L}\sum_{m=-\ell}^{\ell} \mathbf{x}^{(\ell)}_m N_{\ell,m} P^m_\ell(\cos(\theta_j)) cs_m(\varphi_k) \tag{28}$$

where $N_{\ell,m}$ is some normalization factor, P^m_ℓ are the associated Legendre polynomials, and

$$cs_m(\varphi) = \begin{cases} \sin(|m|\varphi) & m < 0 \\ 1 & m = 0 \\ \cos(m\varphi) & m > 0 \end{cases}. \tag{29}$$

We note that we can first evaluate

$$g_m(\theta_j) = \sum_{\ell=0}^{L} \mathbf{x}^{(\ell)}_m N_{\ell,m} P^m_\ell(\cos(\theta_j)) \tag{30}$$

where we set $P_\ell^m = 0$ if $m > \ell$. If we have a set of irreps up to L then we do the summation and this takes $\mathcal{O}(L)$ time. If we only have one irrep to encode then this takes $O(1)$ time. But we also have $\mathcal{O}(L)$ values of θ_j on the grid and $\mathcal{O}(L)$ values of m to evaluate. This gives $\mathcal{O}(L^3)$ runtime to encode onto the grid for irreps up to L and $\mathcal{O}(L^2)$ for a single irrep. Finally evaluating

$$F_1(\theta_j, \varphi_k) = \sum_{m=-\ell}^{\ell} g_m(\theta_j) cs_m(\varphi_k) \tag{31}$$

for a set of φ_k can be done through a FFT in $\mathcal{O}(L \log L)$ time for each θ_j giving $\mathcal{O}(L^2 \log L)$ total. Hence we see encoding onto the sphere takes $\mathcal{O}(L^3)$ time for irreps up to L and $\mathcal{O}(L^2 \log L)$ time for a single irrep.

For the multiplication of signals, we just have elementwise multiplication $F_3(\theta_k, \varphi_k) = F_1(\theta_k, \varphi_k) \cdot F_2(\theta_k, \varphi_k)$. Since there are $\mathcal{O}(L^2)$ grid points this takes $\mathcal{O}(L^2)$ time.

Finally, we decode the signal back into irreps. To do so we use the fact that

$$\mathbf{f}_m^{(\ell)} = \sum_{j,k} a_j F(\theta_j, \varphi_k) Y_{\ell,m}(\theta_j, \varphi_k) \tag{32}$$

for some coefficients a_j. This is essentially performing numerical integration of our signal against a spherical harmonic. Once again using the factorization of the spherical harmonics we get

$$\mathbf{f}_m^{(\ell)} = \sum_j \left(\sum_k F(\theta_j, \varphi_k) cs_m(\varphi_k) \right) a_j N_{\ell,m} P_\ell^m(\cos(\theta_j)). \tag{33}$$

The inner sum in parentheses can be computed in $\mathcal{O}(L)$ time and we need to compute it for $\mathcal{O}(L^2)$ values of θ_j, m pairs giving a runtime of $\mathcal{O}(L^3)$. Of course, we note that cs really is just sines and cosines so alternatively we can use FFT which takes $\mathcal{O}(L^2 \log L)$ total. Computing the outer sum takes $\mathcal{O}(L)$ since we sum over $\mathcal{O}(L)$ values of j. For a single irrep there are $\mathcal{O}(L)$ values of j giving $\mathcal{O}(L^2)$ for the outer sum. For irreps up to ℓ there are $\mathcal{O}(L^2)$ pairs of ℓ, m giving $\mathcal{O}(L^3)$ runtime for the outer sum. In total, we see going from the grid to the coefficients takes $\mathcal{O}(L^2 \log L)$ for a single irrep and $\mathcal{O}(L^3)$ for all irreps.

However, it turns out that the associated Legendre polynomials have recurrence properties which can be exploited to make transforming a set of irreps up to L to the grid and a set of irreps up to L back from the grid asymptotically more efficient (Healy et al., 2003). The runtime for this algorithm which we will call S2FFT is $\mathcal{O}(L^2 \log^2 L)$.

B.3. Fused Tensor Product

Here we describe and analyze the time complexity of fused tensor product. Let L_1, L_2 be the max ℓ's of the inputs and L_3 be the max ℓ of the outputs. We pick some $\tilde{\ell} = \lceil \max(L_1, L_2, L_3)/2 \rceil$ so that $\tilde{\ell} \otimes \tilde{\ell}$ when decomposed into irreps can contain all irreps of the inputs and outputs. Note in principle we could always choose larger $\tilde{\ell}$.

In the following runtime analysis, we assume $L_1 = L_2 = L$, $\tilde{l} = L$, and $L_3 = 2L$.

B.3.1. Naive runtime

The first step of FTP is to convert our input irreps into a tensor product rep using Clebsch-Gordan coefficients as

$$\mathbf{X}_{m_1,m_2}^{(\ell)} = \sum_{m_3=-\ell}^{\ell} C_{\tilde{\ell},m_1,\tilde{\ell},m_2}^{\ell_3,m_3} \mathbf{x}_{m_3}^{(\ell)} \tag{34}$$

$$\mathbf{Y}_{m_1,m_2}^{(\ell)} = \sum_{m_3=-\ell}^{\ell} C_{\tilde{\ell},m_1,\tilde{\ell},m_2}^{\ell_3,m_3} \mathbf{y}_{m_3}^{(\ell)}. \tag{35}$$

Naively we sum over $\mathcal{O}(L)$ values of m_3 and need to do the computation for $\mathcal{O}(L^2)$ possible pairs of m_1, m_2. This gives $\mathcal{O}(L^3)$ naive runtime for converting a single irrep into a tensor product rep. To do so for all irreps up to L the takes $\mathcal{O}(L^4)$ time.

We can then sum over tensor product reps to create

$$\mathbf{X} = \sum_{\ell} \mathbf{X}^{(\ell)} \qquad \mathbf{Y} = \sum_{\ell} \mathbf{Y}^{(\ell)}. \tag{36}$$

There are $\mathcal{O}(L)$ matrices to sum over if we have irreps up to L. Summing matrices takes $\mathcal{O}(L^2)$ time since our matrices are size $\mathcal{O}(L) \times \mathcal{O}(L)$. Hence, this takes $\mathcal{O}(L^3)$ time if we have irreps up to L. If we have a single irrep then we do not need to do anything.

We then multiply the matrices giving $\mathbf{Z} = \mathbf{X}\mathbf{Y}$. Using the naive matrix multiplication algorithm requires $\mathcal{O}(L^3)$ runtime.

Finally we can use Clebsch-Gordan to extract individual irreps giving

$$(\mathbf{x} \otimes_{\text{FTP}} \mathbf{y})^{(\ell_3)}_{m_3} = \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C^{(\ell_3,m_3)}_{\ell_1,m_1,\ell_2,m_2} \mathbf{Z}_{m_1,m_2}. \tag{37}$$

Again, naively we sum over $\mathcal{O}(L^2)$ pairs of m_1, m_2 and need to evaluate $\mathcal{O}(L)$ values of m_3 for $\mathcal{O}(L^3)$ conversion for single irrep. If we want all irreps up to $2L$ then we need $\mathcal{O}(L^4)$.

B.3.2. Optimized runtime with sparsity

Similar to the CGTP, we can take sparsity of the Clebsch-Gordan coefficients into account. We have nonzero values only if $\pm m_1 \pm m_2 = m_3$. Hence in the encoding step, for fixed m_1, m_2 we only need to sum over constant number of m_3 instead of $\mathcal{O}(L)$. This gives a reduction of L in encoding to tensor product rep. Similarly in the decoding step, we see for fixed m_3 we only need to sum over $\mathcal{O}(L)$ pairs of m_1, m_2. This gives a reduction of L as well in decoding back into irreps.

C. Simulating the Fully-Connected Clebsch-Gordan Tensor Product with Gaunt Tensor Products

One way to increase the expressivity of GTP is to first reweight the inputs $\mathbf{x}, \mathbf{y}$. That is, we first create

$$\mathbf{x}'^{(\ell)} = a_\ell \mathbf{x}^{(\ell)} \tag{38}$$

$$\mathbf{y}'^{(\ell)} = b_\ell \mathbf{y}^{(\ell)}. \tag{39}$$

where a_ℓ and b_ℓ are learnable weights. We then perform GTP after this reweighting and extract some output irrep(s) ℓ_3. That is we get

$$(\mathbf{x}' \otimes_{\text{GTP}} \mathbf{y}')^{(\ell_3)}. \tag{40}$$

The analogous operation is fully connected CGTP. There may be multiple pairs of irreps which give a ℓ_3 output. We can always weight and sum these to get

$$\sum_{\ell,\ell'} w_{\ell,\ell'} (\mathbf{x}^{(\ell)} \otimes_{\text{CG}} \mathbf{y}^{(\ell')})^{(\ell_3)} \tag{41}$$

where $w_{\ell,\ell'}$ are learnable weights.

However, even if we only care about symmetric tensor products, the weighted GTP operation is strictly less expressive than fully connected CGTP.

More concretely, suppose we have nontrivial $\ell = 2$ and $\ell = 4$ data in our inputs. From CGTP and the selection rules we see that

$$(\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)} \qquad (\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)} \tag{42}$$

$$(\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)} \qquad (\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)} \tag{43}$$

are all nonzero. In particular, it is possible to create a $\ell = 2$ output of

$$(\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)} + (\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)}$$

with a fully connected CGTP. However, GTP instead gives a single $\ell = 2$ output of form

$$c^2_{2,2}(\mathbf{x}'^{(2)} \otimes_{\text{CG}} \mathbf{y}'^{(2)})^{(2)} + c^2_{2,4}(\mathbf{x}'^{(2)} \otimes_{\text{CG}} \mathbf{y}'^{(4)})^{(2)} + c^2_{4,2}(\mathbf{x}'^{(4)} \otimes_{\text{CG}} \mathbf{y}'^{(2)})^{(2)} + c^2_{4,4}(\mathbf{x}'^{(4)} \otimes_{\text{CG}} \mathbf{y}'^{(4)})^{(2)} \tag{44}$$

where the c's are nonzero coefficients. Note that in order to have nonzero $(\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)}$ and $(\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)}$ contributions, a_2, b_2, a_4, b_4 must all be nonzero. However, that means we must have nonzero $(\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)}$ and $(\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)}$ contributions. Therefore weighted GTP is not expressive enough to output $(\mathbf{x}^{(2)} \otimes_{\text{CG}} \mathbf{y}^{(2)})^{(2)} + (\mathbf{x}^{(4)} \otimes_{\text{CG}} \mathbf{y}^{(4)})^{(2)}$, as it will necessarily mix additional terms.

D. Details of Message-Passing Network

Algorithm 1 LEARNABLETENSORPRODUCT

```
Input: Tensor Product $\otimes$, Number of Channels $C$ (for Gaunt tensor product).
  procedure LEARNABLETP($\mathbf{x}_1, \mathbf{x}_2$)
    if $\otimes = \otimes_{\text{CG}}$ then
      return LINEAR($\mathbf{x}_1 \otimes_{\text{CG}} \mathbf{x}_2$)
    if $\otimes = \otimes_{\text{GTP}}$ then
      for $i = 1, 2, \ldots, C$ do
        $\mathbf{x}_1^{(i)} \leftarrow \text{LINEAR}_1^{(i)}(\mathbf{x}_1)$
        $\mathbf{x}_2^{(i)} \leftarrow \text{LINEAR}_2^{(i)}(\mathbf{x}_2)$
        $\mathbf{x}_o^{(i)} \leftarrow \text{LINEAR}_o^{(i)}(\mathbf{x}_1^{(i)} \otimes_{\text{GTP}} \mathbf{x}_2^{(i)})$
      return CONCATENATE($\{\mathbf{x}_o^{(i)} \mid i \in \{1, 2, \ldots, C\}\}$)
  return LearnableTP
```

Algorithm 2 Operation of our Message Passing Neural Network

```
Input: Graph $G$, Message Passing Iterations $T$, Cutoff $d_{\max}$, Spherical Harmonic Degree $\ell$, Tensor Product $\otimes$
  Compute neighbor lists for each node in $G$:
      $(u, v) \in E \iff \|\mathbf{r}_u - \mathbf{r}_v\| \leq d_{\max}$
  Create LEARNABLETENSORPRODUCT from $\otimes$.
  for $v \in V$ do:
    $h_v^{(0)} \leftarrow [1]$
  for $t = 1, 2, \ldots, T$ do:
    for $v \in V$ do:
      $h_v^{(t)} \leftarrow \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \text{MLP}(\|\mathbf{r}_u - \mathbf{r}_v\|) \times \text{LEARNABLETENSORPRODUCT}(h_u^{(t-1)}, Y_\ell(\mathbf{r}_u - \mathbf{r}_v))$
      $h_v^{(t)} \leftarrow \text{GATE}(h_v^{(t)})$
      $h_v^{(t)} \leftarrow \text{CONCATENATE}([h_v^{(t-1)}, h_v^{(t)}])$
      $h_v^{(t)} \leftarrow \text{LINEAR}(h_v^{(t)})$
  return $\{h_v^{(T)}\}_{v \in V}$
```

In 1, we create learnable (ie, parametrized) variants of the purely functional tensor products. For the Clebsch-Gordan tensor product $\otimes_{\text{CG}}$, we simply add a linear layer to its output. For the Gaunt tensor product $\otimes_{\text{GTP}}$, we create multiple channels, perform the tensor product channel-wise and then concatenate all irreps. This allows the output to have irreps of multiplicity > 1, even with the Gaunt tensor product. We set the number of channels C as 4 in all experiments with the Gaunt tensor product.

In 2, we use these learnable tensor products in a simple message-passing network, very similar to NequIP (Batzner et al., 2022).

E. Additional Benchmarks

Wall-Clock Time: The elapsed time after compiling using `jax.jit`. To enable accurate measurements we calculate the mean wall-clock time for 100 rounds while performing 10 warmup rounds.

Bandwidth and Throughput: We used `Nsight Compute 2024.2.0.0 build 34181891` for profiling and reported Peak GB/s and GFLOP/s from the individual kernel measurments using Roofline Hierarchical Analysis (Yang, 2020).

GPU: We gathered the GPU plots on an NVIDIA RTX A5500, running the CUDA driver version 550.90.07 and CUDA toolkit version 12.5. We use version 0.4.30 for `jax` and `jaxlib`.

CPU: The CPU plots were gathered were on an AMD EPYC 7313.

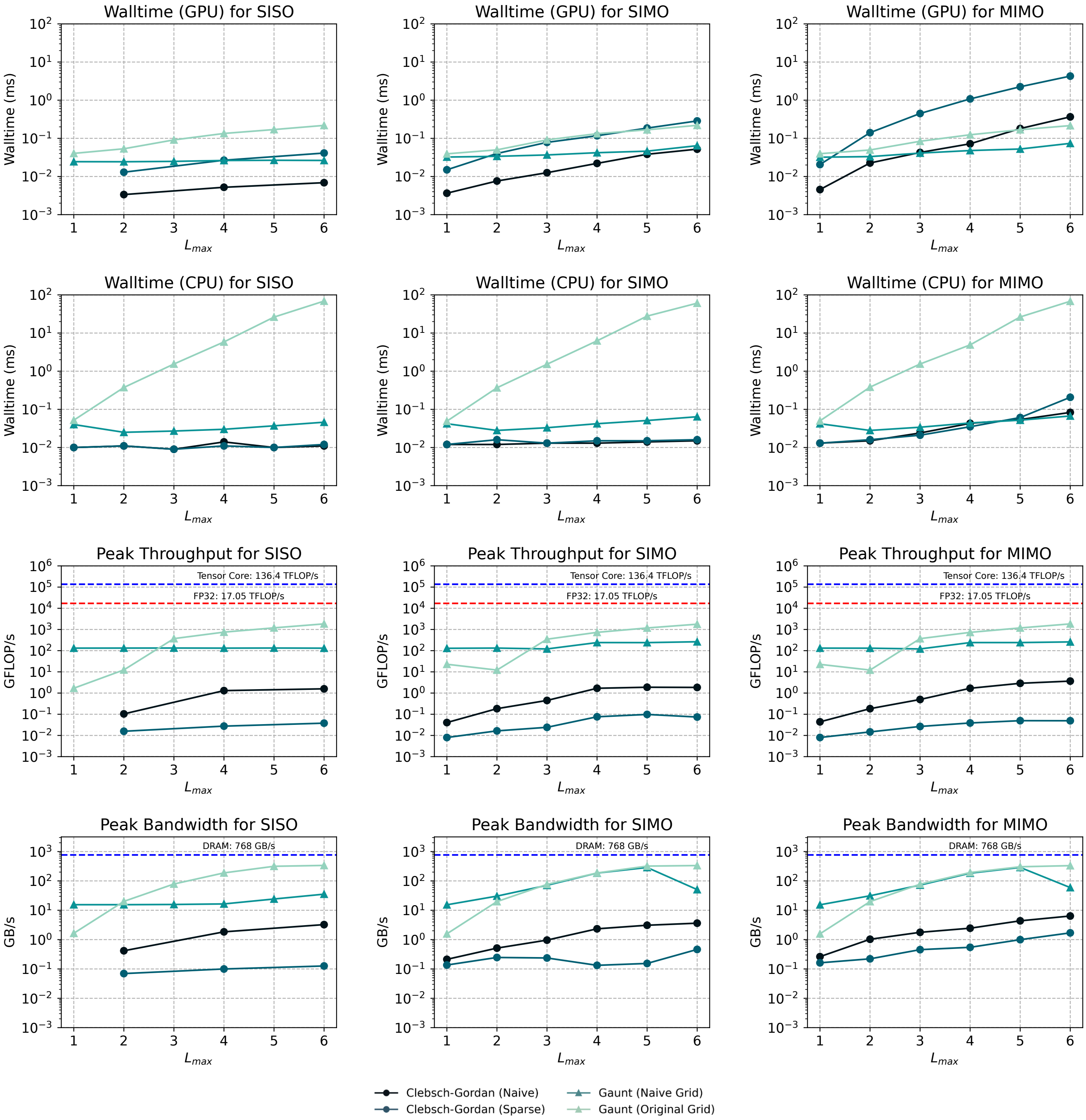

Figure 5. Analysis of SISO, SIMO and MIMO performance for different tensor products: RTX A5500 Walltime (top row), AMD EPYC 7313 Walltime (second row), Peak Throughput (third row) and Peak Bandwidth (bottom row). We had to skip some SISO L_{max} values due to profiling errors.