Towards Flexible, Efficient, and Effective Tensor Product Networks

Nanxiang Wang **Chen Lin** **Michael Bronstein** **Philip Torr**
University of Oxford, UK
nxwang.jeff@gmail.com
{chen.lin, philip.torr}@eng.ox.ac.uk
michael.bronstein@cs.ox.ac.uk

Abstract

Geometric graph neural networks have showcased exceptional performance in modelling geometric data. These models rely heavily on equivariant operations, encompassing vital techniques such as scalarization and the Clebsch-Gordan tensor product. However, tensor-product-based architectures face substantial computational challenges as the representation order increases, significantly limiting their versatility. Moreover, the interpretability of interactions between steerable components remains elusive. In contrast, scalarization methods benefit from cost-efficient invariant scalar operations while still being capable of outperforming certain tensor-product-based models. To bridge the gap between these approaches, we introduce a conceptual framework that emphasizes the potential flexibility in designing tensor product networks. To provide guidance for efficient framework design and gain deeper insights into steerable components, we conduct a preliminary investigation by pruning tensor product interactions. This approach enables us to directly assess the redundancy and significance of steerable components, paving the way for efficient and effective designs.

1 Introduction

Driven by numerous problems in Physics and Chemistry that incorporate geometric data with known symmetry information, there has been a rising interest in constructing equivariant geometric graph neural networks (Cohen and Welling, 2016; Kondor, 2018; Cohen et al., 2019; Jing et al., 2021). By integrating the inductive bias of symmetry into the design of the model, equivariant geometric GNNs consistently demonstrate superior generalization performance and enhanced data efficiency. The most common symmetries in atomistic systems are described by the 3D Euclidean group E(3) and its subgroups such as SE(3), SO(3), and O(3). Among numerous building blocks of E(3)-equivariant geometric GNNs, scalarization and the Clebsch-Gordan (CG) tensor product have garnered sustained interest.

Equivariant architectures that employ the CG tensor product typically convolve the input signal with a learnable kernel basis function such as spherical harmonics (Thomas et al., 2018; Weiler et al., 2018; Fuchs et al., 2020; Brandstetter et al., 2022; Liao and Smidt, 2023). Such formulation generalizes convolutional neural networks via stronger weight tying. Despite notable achievements, two essential problems persist:

- The CG tensor product often entails prohibitive computational costs due to the massive amount of high-dimensional matrix multiplications.
- The interpretability of steerable components in the tensor product remains a puzzle.

As noted by Passaro and Zitnick (2023), tensor product layers using irreducible representations up to order L can entail a computational complexity of $O(L^6)$. While Passaro and Zitnick (2023) introduced a computational trick to reduce the cost from $O(L^6)$ to $O(L^3)$ by realigning the axis of

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

embeddings, the cost of $O(L^3)$ remains substantial. This cost is further exacerbated by the subsequent fully connected linear layers mapping across different channels. Moreover, the understanding towards how steerable components contribute to the learning process remains limited. For instance, questions persist about whether lower-order representations are more effective than higher-order ones in certain tasks and whether intra-order interactions are more important than cross-order interactions.

On the other hand, scalarization is characterized by the versatile use of invariant scalar functions in the update procedure (Schütt et al., 2017; Klicpera et al., 2020b,a; Gasteiger et al., 2021; Satorras et al., 2021; Liu et al., 2022). It is a common belief that nonlinear message exchanges across steerable components of different orders yield better results compared to simple message exchanges within the same order. However, the success of scalarization challenges this assumption. EGNN employs a simple scalarization approach that multiplies coordinate embeddings by invariant scalar functions (Satorras et al., 2021). Despite its simple update procedure, EGNN outperforms various tensor product models across a range of datasets. While this intriguing phenomenon is partly justified by the universality of invariant scalar functions (Villar et al., 2021), few studies have exploited this result to re-evaluate the design of tensor product networks.

Since there are situations where steerable features are essential, such as the QH9 dataset (Yu et al., 2023), we cannot circumvent the problem by solely resorting to scalarization. Therefore, a desirable approach is to establish an intermediate framework that bridges the gap between scalarization and the CG tensor product. To this end, our objective is not to propose a deterministic architecture but to lay the foundation for a more flexible and customizable framework. Leveraging the universal property of the tensor product, we introduce a conceptual framework to highlight the flexibility in the potential design space of tensor product networks (Sec. 4). By drawing connections between scalarization and a typical CG tensor product framework (Sec. 5), we interpret the success of scalarization as conveying two messages: (1). complex interactions across steerable components of different orders might be redundant; (2). non-universal approximators with effective inductive bias might outperform universal approximators with no inductive bias. These insights naturally prompt the consideration of sparsification at the tensor product layer. However, the challenge of interpreting steerable components underscores the need for thoughtful guidance in the design of sparsification.

In pursuit of better efficiency and deeper understanding, we employ L1 loss and pruning techniques as instruments for assessing the redundancy in cross-order interactions and measuring the relative significance of steerable components. Pruning can be seen as an effort to integrate tensor product networks towards scalarization. Moreover, the pruning procedure is customizable for every architecture, allowing different insights to be drawn. Our work is the first to investigate the role of steerable components through pruning and measure the importance of tensor product interactions using sparsity scores. Our experiment demonstrates that, depending on the settings, 40% to 87% of the tensor product interactions can be removed without compromising performance, revealing the existence of redundancy. Additionally, we discover that intra-order interactions with scalar outputs tend to be more significant than others, offering insights for future design considerations. Our experiment code is available at `https://github.com/NW-JEFF/Tensor-Product-Pruning`.

2 Preliminaries

Group representations. A *group representation* $\rho : G \to \mathrm{GL}(V)$ is a homomorphism from a group G to the space of linear bijections on a vector space V. A representation is *irreducible* if the only subrepresentations are itself and its restriction to the vector space of zero. Crucially, for any representation $\rho : \mathrm{SO}(3) \to \mathrm{GL}(\mathbb{R}^d)$, there exists an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$ and irreducible representations $\mathbf{D}^{(l_i)} : \mathrm{SO}(3) \to \mathrm{GL}(\mathbb{R}^{2l_i+1})$ such that

$$\forall R \in \mathrm{SO}(3), \rho(R) = \mathbf{Q}^T \left(\bigoplus_i \mathbf{D}^{(l_i)}(R) \right) \mathbf{Q},$$

where $d = \sum_i (2l_i + 1)$ with $l_i \in \mathbb{N}$ and $\oplus$ denotes the direct sum (block-diagonal concatenation for matrices). The matrices $\mathbf{D}^{(l_i)}(R) \in \mathbb{R}^{(2l_i+1)\times(2l_i+1)}$ are orthogonal and are referred to as the *Wigner-D matrices*, indexed with their *types* (or *orders*) l_i.

Steerable components. A vector space $V = \mathbb{R}^d$ is called *steerable* if it is equipped with a representation of SO(3); elements $\tilde{\mathbf{v}} \in V$ are referred to as *steerable vectors* and we denote them with a tilde for emphasis. Since each $\mathbf{D}^{(l_i)}$ in the direct sum $\bigoplus_i \mathbf{D}^{(l_i)}$ acts on independent components of

$\mathbb{R}^d$, they essentially split $V = \mathbb{R}^d$ into $\bigoplus_i V_{l_i} = \bigoplus_i \mathbb{R}^{2l_i+1}$. In particular, for $\tilde{\mathbf{v}} = \bigoplus_i \tilde{\mathbf{v}}_i \in \bigoplus_i V_{l_i}$, $\tilde{\mathbf{v}}_i \in V_{l_i}$ is called a *type-l_i steerable component* of $\tilde{\mathbf{v}}$. We will index the entries of type-l steerable vectors and $\mathbf{D}^{(l)}$ using $\{-l, ..., l\}$ instead of $\{1, ..., 2l+1\}$.

Equivariance. Let $\mathcal{X}$ and $\mathcal{Y}$ be vector spaces and G be a group. Given two representations $\rho_{\text{in}} : G \to \text{GL}(\mathcal{X})$ and $\rho_{\text{out}} : G \to \text{GL}(\mathcal{Y})$, a function $f : \mathcal{X} \to \mathcal{Y}$ is *G-equivariant* if

$$\forall x \in \mathcal{X}, g \in G,\ f(\rho_{\text{in}}(g)x) = \rho_{\text{out}}(g)f(x).$$

Specially, f is *G-invariant* if $\forall x \in \mathcal{X}, g \in G,\ f(\rho_{\text{in}}(g)x) = f(x)$.

Clebsch-Gordan tensor product. Let $V_l = \mathbb{R}^{2l+1}$ denote a type-l steerable vector space. The *Clebsch-Gordan (CG) tensor product* is defined by

$$\otimes_{\text{CG}} : V_{l_1} \times V_{l_2} \to V_{l_1} \otimes V_{l_2},\ (\tilde{\mathbf{h}}^{(l_1)}, \tilde{\mathbf{h}}^{(l_2)}) \mapsto \mathbf{C}^{(l_2,l_1)} \left(\tilde{\mathbf{h}}^{(l_1)} \otimes \tilde{\mathbf{h}}^{(l_2)}\right),$$

where $V_{l_1} \otimes V_{l_2}$ denotes the tensor product space with $\otimes$ being realized as the Kronecker product, and $\mathbf{C}^{(l_2,l_1)} \in \mathbb{R}^{(2l_2+1)(2l_1+1)\times(2l_2+1)(2l_1+1)}$ is referred to as the *Clebsch-Gordan (CG) coefficients*. $\mathbf{C}^{(l_2,l_1)}$ is the orthogonal change-of-basis matrix such that

$$\forall R \in \text{SO(3)}, \mathbf{D}^{(l_2)}(R) \otimes \mathbf{D}^{(l_1)}(R) = \left(\mathbf{C}^{(l_2,l_1)}\right)^T \left(\bigoplus_{J=|l_2-l_1|}^{l_2+l_1} \mathbf{D}^{(J)}(R)\right) \mathbf{C}^{(l_2,l_1)},$$

As a result, using the relations $\mathbf{u} \otimes \mathbf{v} = \text{vec}(\mathbf{u}\mathbf{v}^T)$ and $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$ for any vectors $\mathbf{u}, \mathbf{v}$ and conformable matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, it is easy to show that the CG tensor product satisfies the following equivariance:

$$\left(\mathbf{D}^{(l_1)}(R)\tilde{\mathbf{h}}^{(l_1)}\right) \otimes_{\text{CG}} \left(\mathbf{D}^{(l_2)}(R)\tilde{\mathbf{h}}^{(l_2)}\right) = \left(\bigoplus_{J=|l_2-l_1|}^{l_2+l_1} \mathbf{D}^{(J)}(R)\right) \left(\tilde{\mathbf{h}}^{(l_1)} \otimes_{\text{CG}} \tilde{\mathbf{h}}^{(l_2)}\right).$$

The above can be easily adapted from SO(3) to O(3) by noticing that the orthogonal irreducible representations of O(3) over the difference set O(3)\SO(3) are either identical to the Wigner-D matrices or the sign-flipped versions of them. For convenience, we will refer to them as the Wigner-D matrices with *even* and *odd* parities, respectively. Accordingly, we append parity $p \in \{o, e\}$ into type statements of steerable vectors, where o stands for odd and e stands for even. For example, any vector acted by $\mathbf{D}^{(l)}$ with odd parity is of type (l, o). After updating the Wigner-D matrices with parities, the definition and properties of the CG tensor product remain the same for O(3), except for one additional rule: the output parity of $\otimes_{\text{CG}}$ is determined by the multiplication law given by

$$e \times e \to e,\ o \times o \to e,\ e \times o \to o,\ \ o \times e \to o.$$

Tensor product layers. In a typical tensor product layer, for every combination of input and output types $(l_{\text{in1}}p_{\text{in1}}, l_{\text{in2}}p_{\text{in2}}, l_{\text{out}}p_{\text{out}})$, each multiplicity of channels is associated with a different learnable weight. The tuple $(l_{\text{in1}}p_{\text{in1}}, l_{\text{in2}}p_{\text{in2}}, l_{\text{out}}p_{\text{out}})$ is referred to as the *path* of the interaction. In particular, we refer to the interaction as *intra-order* when $l_{\text{in1}} = l_{\text{in2}}$ and *cross-order* when $l_{\text{in1}} \neq l_{\text{in2}}$.

3 Related work

Tensor products. Tensor field network (TFN) stands as one of the earliest models that rely on the CG tensor product to construct SE(3)-equivariant networks (Thomas et al., 2018). Within their work, tensor products emerge as a means to preserve equivariance between layers operating on point clouds. Weiler et al. (2018) more explicitly deduce the same kernel parameterization by solving the equivariance constraint involving tensor products. Beyond pure convolution, SE(3)-Transformer adds invariant self-attention weights before TFN's convolution kernel (Fuchs et al., 2020), interpolating between graph attention networks and TFN. On the other hand, Equiformer integrates the CG tensor product towards the original Transformer by replacing operations from the original Transformer with their E(3)-equivariant counterparts (Liao and Smidt, 2023). To allow complicated nonlinear interactions, SEGNN extends the geometric message passing framework by employing steerable embeddings and replacing standard MLP with steerable MLP (Brandstetter et al., 2022).

Our proposed conceptual tensor product framework highlights the potential flexibility in designing the input and output of the tensor product layers, which unifies the aforementioned designs. Additionally, we provide interpretation for high-order steerable components, a previously unexplored territory.

Network pruning. Our approach to evaluating the redundancy and relative importance of high-order steerable components draws inspiration from MLP pruning (Han et al., 2015) and the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2019). Han et al. (2015) propose to use a three-step method (train, prune, and retrain) to learn network connections, and LTH states that there exists a sparse subnetwork in a densely initialized model that can be trained to match full model performance. While we follow a similar three-step method and confirm the applicability of the Lottery Ticket Hypothesis in our settings, our primary focus lies in providing guidance for designing effective tensor product networks.

Numerous strategies have been proposed to enhance efficient learning specifically on GNNs (Ying et al., 2019; Zhou et al., 2021; Chen et al., 2021; Lutzeyer et al., 2022; Peng et al., 2022; Liu et al., 2023). Existing strategies involve pruning nodes, edges, channels, weights, or a combination of them, but none of the them investigates pruning tensor product interactions. In addition, beyond utilizing pruning to improve efficiency or transferring LTH to another unexplored setting, our primary focus is to offer interpretations for tensor product interactions.

4 Exploring the design space of tensor products

Now, we lay down a foundational rationale for considering tensor products in the construction of equivariant layers. The insight is drawn from a basic fact known as the universal property of the tensor product (Procesi, 2006).

The Universal Property. A tensor product on vector spaces V and W is a bilinear map from $V \times W$ to a vector space denoted as $V \otimes W$,

$$\otimes : V \times W \to V \otimes W, \ (v, w) \mapsto v \otimes w,$$

such that for every vector space Z and every bilinear map $h : V \times W \to Z$, there uniquely exists a linear map $\hat{h} : V \otimes W \to Z$ such that $h = \hat{h} \circ \otimes$. The tensor product $\otimes$ and the vector space $V \otimes W$ uniquely exist up to isomorphism.

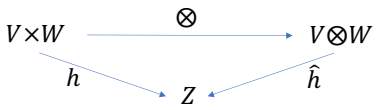

Figure 1: Commute diagram illustrating the universal property of the tensor product.

As illustrated in Fig. 1, the tensor product $\otimes$ on $V \times W$ can be seen as an intermediary such that any bilinear function from $V \times W$ to Z can be uniquely transferred to some linear function from $V \otimes W$ to Z. Since any linear function can be seen as a special bilinear function with the other input fixed (arbitrarily), we can shift our design target from linear equivariant functions to bilinear equivariant functions. Next, the universal property allows working with tensor products (equivariance guaranteed via tensor product representation) and linear equivariant functions on the tensor product space. While this framework seems to complicate matters by taking an unnecessary detour, it does lead to many added benefits:

- Linear equivariant functions on the tensor product space can sometimes be more convenient to parameterize than those on the original space. For example, Pearce-Crump (2023) directly identifies a spanning set of matrices for linear equivariant functions on tensor power spaces of $\mathbb{R}^d$.
- It enhances flexibility in architecture design by allowing an extra level of freedom in choosing how the other input interacts within the architecture.
- It unifies existing tensor product designs under one common framework. For instance, SEGNN's fully connected tensor product layers adopt the CG tensor product with one input fixed as the spherical harmonics, followed by fully connected linear mappings at the level of individual steerable components. Besides, augmenting the linear mapping by non-linear invariant weights recovers self-attention and radial bases.

We strive to draw attention to the fact that this conceptual framework can be flexibly adapted to cater to varying needs. As we will demonstrate in the next section, we may replace spherical harmonics with a concatenation of steerable features to achieve a balance between efficiency and efficacy.

5 Sparsifying tensor product layers

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let each node $v_i \in \mathcal{V}$ be associated with a 3D coordinate $\mathbf{x}_i$ and a steerable feature $\tilde{\mathbf{f}}_i$ in $\mathbb{R}^d$. The node update procedures of EGNN and TFN can be rephrased as

$$\text{EGNN: } \mathbf{x}_i' = \mathbf{x}_i + \sum_{j \in \mathcal{N}(i) \setminus \{i\}} \alpha(||\mathbf{x}_i - \mathbf{x}_j||^2)(\mathbf{x}_i - \mathbf{x}_j),$$

$$\text{TFN: } \tilde{\mathbf{f}}_i' = \sum_{j \in \mathcal{N}(i)} \mathbf{W}_{\tilde{\mathbf{a}}_{ij}}(||\mathbf{x}_i - \mathbf{x}_j||^2)\tilde{\mathbf{f}}_j,$$

where $\alpha(\cdot)$ is a scalar function and the mapping $\tilde{\mathbf{f}}_i \mapsto \mathbf{W}_{\tilde{\mathbf{a}}_{ij}}(||\mathbf{x}_i - \mathbf{x}_j||^2)\tilde{\mathbf{f}}_i$ represents a CG tensor product layer parameterized by learnable weights that depend on $||\mathbf{x}_i - \mathbf{x}_j||^2$: the tensor product involves $\tilde{\mathbf{f}}_i$ and the spherical harmonics embedding

$$\tilde{\mathbf{a}}_{ij} := \bigoplus_{l \geq 0} \tilde{\mathbf{Y}}^{(l)}\left(\frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_j - \mathbf{x}_i||}\right), \text{ where } \tilde{\mathbf{Y}}^{(l)} = \begin{pmatrix} Y_{-l}^{(l)} \\ \vdots \\ Y_{l}^{(l)} \end{pmatrix}, \; Y_m^{(l)} : S^2 \to \mathbb{R}, \; \forall m \in \{-l, ..., l\}.$$

The distinction between the two models in the use of $\tilde{\mathbf{f}}_j$ and $(\mathbf{x}_i - \mathbf{x}_j)$ can be ignored for our purpose, as they handle different equivariant targets. Two observations can be made from the above contrast: (1). $\mathbf{W}_{\tilde{\mathbf{a}}_{ij}}(||\mathbf{x}_i - \mathbf{x}_j||^2)$ is a matrix; (2). the scalar $\alpha(||\mathbf{x}_i - \mathbf{x}_j||^2)$ can be equivalently seen as $\alpha(||\mathbf{x}_i - \mathbf{x}_j||^2)\,\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. By comparing the formulation of $\alpha(||\mathbf{x}_i - \mathbf{x}_j||^2)\,\mathbf{I}$ and $\mathbf{W}_{\tilde{\mathbf{a}}_{ij}}(||\mathbf{x}_i - \mathbf{x}_j||^2)$, we notice that EGNN's update layer resembles a sparse version of TFN's tensor product layers. From this respect, we interpret the phenomenon of EGNN outperforming TFN as suggesting that complicated cross-order interactions may be redundant.

In addition, DimeNet++ manually extracts relative distances and inter-atomic angles to account for directional geometric information (Klicpera et al., 2020a), resulting in even better performance when predicting invariant targets. While propagating solely a manual extraction of scalar features results in the loss of certain geometric information, DimeNet++ still massively outperforms TFN, which is theoretically universal (Dym and Maron, 2021). One interpretation is that those manually extracted features encode a geometric inductive bias. Such a bias might enhance performance when it aligns with the underlying reality, even if universality is not attained.

These insights directly motivate two strategies designed to boost efficiency:

1. Removing redundant tensor product interactions (which we address in Sec. 6).
2. Redesigning the tensor product layer to incorporate better inductive bias.

To illustrate the second strategy, we will generalize the inductive bias adopted by EGNN by referring to the conceptual framework outlined in Sec. 4. Note that EGNN only propagates the relative distance between two nodes within the message function: $||\mathbf{x}_i - \mathbf{x}_j||^2 \equiv \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$. Transferring to steerable features, this is analogous to using a linear combination of the scalar tuple $\left(\tilde{\mathbf{f}}_i^T\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j^T\tilde{\mathbf{f}}_j, \tilde{\mathbf{f}}_i^T\tilde{\mathbf{f}}_j\right)$. Suppose $\tilde{\mathbf{f}}_i$ splits into steerable components $\{\tilde{\mathbf{f}}_i^{(l)}\}_l$. We may generalize by considering the effects of those channels separately:

$$\left(\{\tilde{\mathbf{f}}_i^{(l)\,T}\tilde{\mathbf{f}}_i^{(l)}\}_l, \{\tilde{\mathbf{f}}_j^{(l)\,T}\tilde{\mathbf{f}}_j^{(l)}\}_l, \{\tilde{\mathbf{f}}_i^{(l)\,T}\tilde{\mathbf{f}}_j^{(l)}\}_l\right).$$

This inductive bias can be seen as an instance of our tensor product network as follows: each tensor product takes two copies of $\tilde{\mathbf{f}}_i \oplus \tilde{\mathbf{f}}_j$ as inputs, forms the CG tensor product $\left(\tilde{\mathbf{f}}_i \oplus \tilde{\mathbf{f}}_j\right) \otimes_{\text{CG}} \left(\tilde{\mathbf{f}}_i \oplus \tilde{\mathbf{f}}_j\right)$, and linearly maps to $\left(\{\tilde{\mathbf{f}}_i^{(l)\,T}\tilde{\mathbf{f}}_i^{(l)}\}_l, \{\tilde{\mathbf{f}}_j^{(l)\,T}\tilde{\mathbf{f}}_j^{(l)}\}_l, \{\tilde{\mathbf{f}}_i^{(l)\,T}\tilde{\mathbf{f}}_j^{(l)}\}_l\right)$. Nevertheless, the computational cost is merely $O(L^2)$ (due to the inner product) in contrast to the original cost of $O(L^6)$.

One may further augment the learning process by sending the output scalar tuple into a nonlinear mapping, such as an MLP, to learn cross-order interactions. In this manner, the design differs from the

original tensor product in that it replaces the spherical harmonics embedding in one of the inputs with another copy of feature concatenation; it also diverges from EGNN in that cross-order interactions can be learned. We hope that our conceptual schemes provide new insight that facilitates more flexible designs.

6 Pruning tensor product interactions

Absorbing insight from MLP pruning (Han et al., 2015) and the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2019), we detect redundancy in tensor product interactions through a self-supervised pre-training procedure. We compare the baseline with two schemes for pre-training the interaction weights:

- L1 loss with one-shot pruning: training with L1 loss and pruning once at the end.
- Global unstructured iterative magnitude pruning (IMP).

Since winning tickets typically occur in the early iterations (Tanaka et al., 2020), we only pre-train for a small number of epochs, so the pre-training schemes do not impose too much additional cost.

If the performance on the test set does not deteriorate when certain tensor product interactions are removed, then the corresponding interactions might be redundant. Since the proposed pruning schemes tend to drive useless weights towards zero, we can measure the importance of each type of interaction based on their relative sparsity induced by the pre-training procedure. Furthermore, we can examine the distributions of the learned weights associated with each type of interaction under varying L1 loss to investigate how they behave in response to different levels of compression.

Analogous to LTH, we also compare the baseline with two schemes for retraining given the pruning mask from pre-training: random reinitialization, and reusing the initialization during pre-training. The retraining serves two purposes: (1). verifying that winning tickets can be identified in this setting, so the conclusions also apply to the full model according to LTH; (2). testing the effectiveness of pruning as a self-supervised method to learn customized interaction modes from the dataset.

While our proposed schemes are relatively basic, we stress that our objective extends beyond merely transferring pruning and LTH to another uncharted setting. Instead, we employ pruning as a method to examine the properties of steerable components, hoping to provide guidance towards effective and efficient equivariant designs.

7 Experiments

Baseline choices. We have chosen SEGNN and Equiformer as our baseline models due to their specialized tensor product designs. SEGNN leverages non-linear message exchanges across steerable components of different orders, which makes it well-suited for evaluating the role of cross-order interactions. Equiformer utilizes MLP attention and depth-wise tensor product layers, offering improved efficiency and an additional inductive bias compared to SEGNN's fully connected tensor product layers. We will use SEGNN as our primary baseline and consider Equiformer for comparison.

While SEGNN has performed ablation studies regarding dimensions of representations, their experiments only tested against the two band limits controlling the maximum orders used. This reveals neither whether fully connected interactions are needed nor whether lower orders are more important than higher orders.

Dataset. We test our experiments on regressing the isotropic polarizability, denoted as α, from the QM9 dataset (Blum and Reymond, 2009; Montavon et al., 2013), which comprises small molecules containing up to 29 atoms. Each atom is represented by its 3D position coordinates and one-hot encoding of its atomic type (H, C, N, O, F). Prediction performance is measured in terms of the mean absolute error (MAE) between predicted values and true values.

Baseline setup. Important parameter choices in the original implementation of SEGNN include

$$l_h = 2, l_a = 3, \text{weight\_decay} = 10^{-8}, \text{epoch} = 1000, \text{scheduler} = \text{MultiStepLR}([0.8, 0.9] * \text{epoch}),$$

where l_a denotes the maximum order in the spherical harmonics embedding and l_h is the band limit for hidden features ($l_h = 2$ means steerable components of types greater than 2 in the hidden features are discarded). It is important to mention that SEGNN maintains an equitable number of weights in

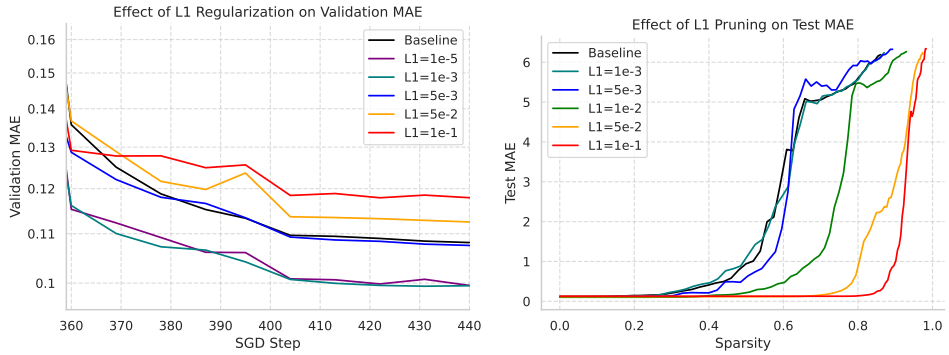

Figure 2: Impact of L1 regularization on validation data (left): the number in the legend indicates the corresponding multiplying factor of the L1 loss. Investigation of the impact of L1 loss on sparsity and test performance (right).

CG tensor product layers, ensuring that larger values of l do not result in extra weights. This enables a fair and direct comparison between different orders.

We pre-train the network using epoch $= 50$. Reproducibility experiments have been done to ensure that predictions with epoch $= 50$ are relatively stable with a standard error of 0.0053. For a fair comparison, other variables are unchanged in our experiments. After full training, the set of model parameters that leads to the lowest validation error is used to evaluate on the test set.

Pruning implementation. Throughout the experiments, we only apply pruning and L1 loss to learnable weights of the fully connected tensor product layers (not including biases). L1 loss with one-shot pruning is implemented by first adding an L1 loss with a multiplying factor (set as a hyperparameter) and then setting weights below a specified threshold to zero after training. We disable weight decay during L1 loss training. The global unstructured IMP is implemented at intervals of x epochs with a maximum of y shots, where diverse choices of x and y are tested; each time, we prune the same percentage among the remaining unpruned weights. Overall, we find that IMP produces similar outcomes as L1 loss with one-shot pruning. Since IMP does not offer additional insights for our objectives, we choose not to present its results.

L1 loss pre-training with one-shot pruning. Fig. 2 (left) allows us to identify key levels of L1 regularization that act as watersheds: L1 loss with a multiplying factor in the range $[0.01, 0.05]$ gives an approximate upper bound that improves or maintains a competitive performance compared to the baseline, and multiplying factors greater than 0.05 lead to poorer performance. To identify a proper pruning threshold, we generate 50 evenly spaced numbers over the interval (0, 0.1) and respectively set weights below these numbers to zero. We then evaluate the test MAE for each of the pruned models. Fig. 2 (right) plots the test MAEs against sparsity. By considering both plots, we determine that L1=0.005 is the optimal level of regularization. Under this setting, the pre-training procedure induces a sparsity of up to 40% without compromising performance, which reveals the presence of redundancy.

Scalability. We repeat the same pre-training procedure for larger orders. For $l_h = 3$, $l_a = 4$, we find that 87% of weights can be pruned without compromising performance compared to the baseline for $l_h = 3$, $l_a = 4$. Moreover, we observe that applying pruning to higher-order settings does not lead to better performance than directly pruning $l_h = 2, l_a = 3$. Since the total number of weights is equitable across different choices of orders, our observations imply that redundancy is more prominent when higher-order components are present. This leads to the speculation that higher-order components may be less effective at encoding geometric information when the prediction target is a scalar.

Lotter ticket hypothesis. To check that our conclusions drawn from the pre-training procedure apply to the full model, we verify LTH by comparing two schemes for retraining the pruned models: random reinitialization, and reusing the initialization during pre-training. Fig. 3 illustrates two

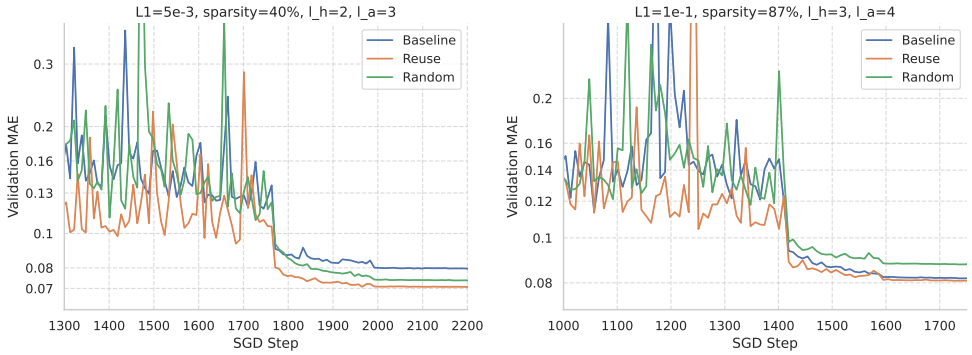

Figure 3: Baseline vs. retrained pruned models using two reinitialization schemes. $l_h = 2$, $l_a = 3$ (left) are retrained for 250 epochs, whereas $l_h = 3$, $l_a = 4$ (right) are retrained for 200 epochs.

phenomena: (1). reusing the initialization from pre-training works better than random reinitialization; (2). reusing the initialization from pre-training enables the pruned network to converge faster than the baseline while not losing accuracy. The occurrence of these two phenomena confirms LTH in our setting, thus justifying the use of pre-training with pruning in diagnosing the full model.

Relative importance of tensor product interactions. We assess the importance of tensor product interactions (conditional on the L1 loss) by calculating the sparsity of the corresponding path when the model is pruned at the maximum threshold such that model performance does not deteriorate. The smaller the sparsity induced by the L1 loss, the more significant the interaction. The relative importance between any two types of interactions can then be computed by the inverse ratio of their sparsity scores. Table 1 presents the smallest five sparsity scores at $l_h = 3$, $l_a = 4$.

Table 1: Top five sparsity scores for SEGNN in ascending order at $l_h = 3$, $l_a = 4$. L1=5×10^{-3} maintains competitive performance as the baseline, L1=5×10^{-2} performs slightly worse than the baseline, and L1=5×10^{-1} performs significantly worse than the baseline.

L1=5×10^{-3}		L1=5×10^{-2}		L1=5×10^{-1}	
Path	Sparsity	Path	Sparsity	Path	Sparsity
$(0e, 0e, 0e)$	0.0789	$(0e, 3o, 3o)$	0.4419	$(0e, 3o, 3o)$	0.8881
$(3o, 3o, 0e)$	0.1001	$(0e, 2e, 2e)$	0.4671	$(0e, 2e, 2e)$	0.9158
$(2e, 2e, 0e)$	0.1002	$(3o, 3o, 0e)$	0.4736	$(2e, 4e, 2e)$	0.9216
$(2e, 0e, 2e)$	0.1135	$(2e, 2e, 0e)$	0.4917	$(1o, 4e, 3o)$	0.9249
$(2e, 4e, 2e)$	0.1190	$(0e, 0e, 0e)$	0.5021	$(2e, 3o, 3o)$	0.9292

Table 1 highlights an interesting pattern: many paths take the form of $(le, le, 0e)$, indicating the significance of intra-order interactions with scalar outputs. As the L1 regularization gets stronger, interactions with higher output orders become more prominent. When the regularization is so strong that performance seriously deteriorates (L1=5×10^{-1}), scalar-output interactions lose their leadership. We interpret this observation as suggesting that intra-order interactions with scalar outputs are naturally more effective than cross-order interactions in encoding important information (at least when the prediction target is a scalar). As a result, modifying the baseline architecture to more effectively leverage intra-order interactions (with scalar outputs) could lead to both higher cost-efficiency and improved performance. This, in turn, motivates the use of designs such as inner products and self-attention.

Meanwhile, we speculate that different orders of steerable components carry varying information capacities that suit different compression rates. High-order components carry more information than low-order ones, but the equivariant constraints render high-order components cost-inefficient, so trained models naturally prefer low-order ones unless there is a restriction on the number of weights.

As a result, mild L1 regularization encourages the network to prioritize low-order components, whereas radical L1 regularization forces the network to resort to high-order ones.

Equiformer. As a comparison, we repeat the experiment on Equiformer, which has a stronger inductive bias compared to SEGNN. The impact of pruning appears less pronounced for Equiformer, with a decline in performance observed even with the removal of just 25% of the weights. This is understandable as Equiformer leverages a more compact design due to depth-wise tensor product layers. Moreover, Table 2 shows that the sparsity scores of intra-order interactions with scalar outputs are massively lower than others for all three levels of L1 regularization. Given Equiformer's heavy reliance on self-attention, this notable disparity in sparsity scores affirms the indispensable role of attention in Equiformer's design. Overall, our diagnosis indicates that Equiformer's design is reasonably effective.

Table 2: Top five sparsity scores for Equiformer in ascending order. L1=5×10^{5} and L1=5×10^{-4} lead to competitive performance as the baseline, whereas L1=5×10^{-3} results in slightly worse outcomes.

L1=5×10^{-5}		L1=5×10^{-4}		L1=5×10^{-3}	
Path	Sparsity	Path	Sparsity	Path	Sparsity
$(1e, 1e, 0e)$	0.0104	$(2e, 2e, 0e)$	0.0208	$(2e, 2e, 0e)$	0.1875
$(2e, 2e, 0e)$	0.0208	$(1e, 1e, 0e)$	0.0234	$(0e, 0e, 0e)$	0.2382
$(0e, 0e, 0e)$	0.0855	$(0e, 0e, 0e)$	0.0979	$(1e, 1e, 0e)$	0.2396
$(1e, 1e, 1e)$	0.1667	$(2e, 1e, 1e)$	0.2031	$(2e, 0e, 2e)$	0.4309
$(2e, 2e, 1e)$	0.1667	$(0e, 2e, 2e)$	0.2101	$(1e, 0e, 1e)$	0.4433

8 Conclusion

We proposed a conceptual framework that highlights the flexibility in the design space of tensor product networks, providing a foundational rationale for considering tensor products in the construction of equivariant layers. We have also revealed the existence of redundancy within tensor product interactions in the baseline model. Our attempt to employ a pruning procedure as a tool for examining redundancy and measuring the relative importance of tensor product interactions serves as an initial step towards a better understanding of their impact on model performance.

Future work. We hope that our conceptual framework and pruning procedure provide valuable insights into the role of tensor product interactions, offering customizable guidance for the efficient and effective design of future tensor product networks. Future directions of work also include conducting experiments to explore the behaviour of tensor product interactions in different settings, such as when the prediction target is a vector.

References

L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and physical quantities improve E(3) equivariant message passing. In *The Tenth International Conference on Learning Representations*, 2022.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1695–1706. PMLR, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999. PMLR, 20–22 Jun 2016.

Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In *Neural Information Processing Systems*, 2019.

Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. In *International Conference on Learning Representations*, 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. In *Advances in Neural Information Processing Systems*, volume 34, pages 6790–6802, 2021.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021.

Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *ArXiv Preprint*, 2020a.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *8th International Conference on Learning Representations*, 2020b.

Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *ArXiv Preprint*, 2018.

Yi-Lun Liao and Tess E. Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

Chuang Liu, Xueqi Ma, Yibing Zhan, Liang Ding, Dapeng Tao, Bo Du, Wenbin Hu, and Danilo P. Mandic. Comprehensive graph gradual pruning for sparse training in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023.

Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *The Tenth International Conference on Learning Representations*, 2022.

Johannes F. Lutzeyer, changmin wu, and Michalis Vazirgiannis. Sparsifying the update step in graph neural networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9): 095003, 2013.

Saro Passaro and C. Lawrence Zitnick. Reducing so(3) convolutions to so(2) for efficient equivariant gnns. In *International Conference on Machine Learning*, 2023.

Edward Pearce-Crump. Brauer's group equivariant neural networks. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27461–27482. PMLR, 2023.

Hongwu Peng, Deniz Gurevin, Shaoyi Huang, Tong Geng, Weiwen Jiang, Orner Khan, and Caiwen Ding. Towards sparsification of graph neural networks. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pages 272–279, 2022.

Claudio Procesi. *Lie Groups: An Approach through Invariants and Representations*. Universitext. Springer New York, 2006.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 9323–9332. PMLR, 2021.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network

for modeling quantum interactions. In *Advances in Neural Information Processing Systems 30*, pages 991–1001, 2017.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389, 2020.

Nathaniel Thomas, Tess E. Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *ArXiv Preprint*, 2018.

Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *Advances in Neural Information Processing Systems*, 2021.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Haiyang Yu, Meng Liu, Youzhi Luo, Alex M Strasser, Xiaofeng Qian, Xiaoning Qian, and Shuiwang Ji. Qh9: A quantum hamiltonian prediction benchmark for qm9 molecules. *ArXiv Preprint*, 2023.

Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. Accelerating large scale real-time gnn inference using channel pruning. *Proc. VLDB Endow.*, 14(9): 1597–1605, 2021.