Multivector Neurons: Better and Faster O(n)-Equivariant Clifford GNNs

Cong Liu [1 2] **David Ruhe** [1 2 3] **Patrick Forré** [1 2]

Abstract

Most current deep learning models equivariant to $\mathrm{O}(n)$ or $\mathrm{SO}(n)$ either consider mostly scalar information such as distances and angles or have a very high computational complexity. In this work, we test a few novel message passing graph neural networks (GNNs) based on Clifford multivectors, structured similarly to other prevalent equivariant models in geometric deep learning. Our approach leverages efficient invariant scalar features while simultaneously performing expressive learning on multivector representations, particularly through the use of the equivariant geometric product operator. By integrating these elements, our methods outperform established efficient baseline models on an N-Body simulation task and protein denoising task while maintaining a high efficiency. In particular, we push the state-of-the-art error on the N-body dataset to 0.0035 (averaged over 3 runs); an 8% improvement over recent methods. Our implementation is available on GitHub.

1. Introduction

Learning on graphs with geometric features has gathered significant attention in the field of geometric deep learning. Particularly, the integration of group equivariance to respect the symmetries in data structures yielded significant research breakthroughs. Among these, $\mathrm{O}(n)$- and $\mathrm{SO}(n)$-equivariance stands out as an advantageous model property when dealing with geometric data from various interdisciplinary scientific fields, such as molecular and protein structures or 3D computer vision (Brandstetter et al., 2022; Liao & Smidt, 2022; Gasteiger et al., 2020; Deng et al., 2021; Gao et al., 2023; Bekkers et al., 2024).

Recently, some works (Brandstetter et al., 2022; 2023; Ruhe et al., 2023b;a; Brehmer et al., 2023; Liu et al., 2024; Zhdanov et al., 2024; Spinner et al., 2024) have designed equivariant models based on the Clifford (or *geometric*) algebra (Ruhe et al., 2023a): Clifford Group Equivariant Neural Networks (CGENNs). These models embed geometric features in the Clifford algebra vector space and, based on linear combinations of *multivectors* (elements of the Clifford algebra) and geometric products, they achieve $\mathrm{O}(n)$- or $\mathrm{SO}(n)$-equivariance. In a sense, CGENNs are similar to standard feedforward neural network but each neuron is multivector-valued. Hence: *Multivector Neurons*. On several benchmarks, CGENNs (in particular, their graph variants) have achieved excellent results. However, their heavily parameterized geometric product layers are computationally intense and consume significant memory.

In this work, we test a novel set of fast and expressive Clifford graph neural networks (GNNs), incorporating parameterizations akin to those used by Equivariant Graph Neural Networks (EGNN) (Satorras et al., 2021), Geometric Vector Perceptron (GVP) (Jing et al., 2021), and Vector Neurons (Deng et al., 2021). In our design choices, instead of relying on expensive geometric product layers in (Ruhe et al., 2023a) for expressiveness, we incorporate more cheap and efficient multi-layer perceptrons operating on invariant scalar features, while ensuring group equivariance through the representation of geometric features as multivectors. Consequently, our approach yields a suite of networks that are both expressive and efficient, making them suitable for large-scale geometric graph applications.

We demonstrate the efficacy of these models in two settings: an N-body simulation task (achieving new state of the art predictive results) and a protein denoising task. The promising results on the protein denoising task makes these models suitable for denoising diffusion generative modeling, which is a promising avenue for future work.

2. Clifford Group Equivariant Neural Networks

CGENNs are based on a powerful mathematical tool: The *Clifford* or *geometric* algebra. The Clifford algebra $\mathrm{Cl}(V, \mathfrak{q})$ is generated by a vector space V coupled with a quadratic form $q : V \to \mathbb{R}$. In this work, we restrict $V := \mathbb{R}^3$ and $\mathfrak{b}$ to be the standard (Euclidean) inner product $b(v) := \langle v, v \rangle$.

[1]AMLab, Amsterdam, Netherlands [2]AI4Science Lab, Amsterdam, Netherlands [3]Anton Pannekoek Institute, Amsterdam, Netherlands. Correspondence to: Cong Liu <c.liu4@uva.nl>.

Accepted as an extended abstract for the Geometry-grounded Representation Learning and Generative Modeling Workshop at the 41st International Conference on Machine Learning, ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

As such, we will write $\mathrm{Cl}(\mathbb{R}^3)$. The Clifford algebra has associated a product, the *geometric product*, which can multiply vectors to generate higher-order objects like bivectors. *Multivectors* $\mathbf{v} \in \mathrm{Cl}(\mathbb{R}^3)$ are general Clifford elements defined as $\mathbf{v} = \sum_{i \in I} c_i v_{i,1} * \cdots * v_{i,k}$, where I is finite, $v_{i,j} \in \mathbb{R}^d$ and $c_i \in \mathbb{R}$. I.e., they are finite linear combinations of geometric products of vectors, denoted with $*$. The Clifford algebra has a bilinear form extended from the usual one $\mathfrak{b} : \mathrm{Cl}(\mathbb{R}^3) \times \mathrm{Cl}(\mathbb{R}^3) \to \mathbb{R}$ and quadratic form $\mathfrak{q}(\mathbf{v}) := \mathfrak{b}(\mathbf{v}, \mathbf{v})$.

The Clifford algebra can be decomposed into orthogonal subspaces. Among these spaces, we have the scalars $\mathbb{R}$ and also the original vector space $\mathbb{R}^3$. As such, Ruhe et al. (2023a) embed vector- or scalar-valued data in the algebra. Then, the data gets processed as multivectors through linear combinations and geometric products using carefully designed neural network layers. In general, these layers are $\mathrm{O}(n)$-equivariant. In this work, we make a sharper distinction between scalars and multivector-valued data, processing the scalars for fast and expressive representation learning, while updating multivectors to maintain the ability to build sophisticated geometric representations. We do so by taking inspiration from established equivariant architectures.

3. Methods

3.1. Clifford-EGNN

Our first tested architecture extends EGNN (Satorras et al., 2021). We consider a graph where each node i is associated with a position $x_i \in \mathbb{R}^3$ and node feature $h_i \in \mathbb{R}^c$. EGNNs take l-th layer node embeddings h_i^l and node positions $x_i^l \in \mathbb{R}^3$ to output $l+1$-th layer node embeddings and positions. The message passing scheme is as follows:

$$m_{ij}^l = \phi_e(h_i^l, h_j^l, ||x_i^l - x_j^l||) \tag{1}$$

$$m_i^l = \sum_{j \in \mathcal{N}_i} m_{ij}^l \tag{2}$$

$$x_i^{l+1} = x_i^l + \sum_{j \in \mathcal{N}_i} \phi_x(m_{ij}^l) \cdot (x_i^l - x_j^l) \tag{3}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i^l) \tag{4}$$

with ϕ_e, ϕ_x, ϕ_h as multi-layer perceptrons acting as edge message networks, position message networks and node features update networks, where $\mathcal{N}_i$ represents the set of neighbor nodes to node i.

In the Clifford counterpart (details decribed below), we use invariant messages to scale equivariant features while processing arrays of multivectors in a feedforward manner. We have several multivector channels that interchange information with a scalar network. In update steps, we apply equivariant geometric product layers on multivectors to refine the multivector features and exchange geometric information multiplicatively.

After an initial linear embedding layer, we can consider an array of multivectors $\mathbf{v} \in \left(\mathrm{Cl}(\mathbb{R}^3)\right)^c$, where c denotes the number of channels. Furthermore, we have scalar features $h \in \mathbb{R}^s$. The message passing scheme from node j to node i is defined as follows.

$$\mathbf{v}_{ij}^l = \boldsymbol{\phi}_e^l(\mathbf{v}_i^l - \mathbf{v}_j^l) \tag{5}$$

$$s_{ij}^l = \left[h_i^l, h_j^l, \mathfrak{q}(\mathbf{v}_{ij}^l)\right] \tag{6}$$

$$m_{ij}^l = \phi_e^l(s_{ij}^l) \tag{7}$$

$$m_i^l = \sum_{j \in \mathcal{N}_i} m_{ij}^l \tag{8}$$

$$\mathbf{v}^{l+1} = \mathbf{v}_i^l + \frac{1}{\sqrt{|\mathcal{N}_i|}} \boldsymbol{\psi}_v^l \left(\sum_{j \in \mathcal{N}_i} \phi_v^l(m_{ij}^l) \cdot \mathbf{v}_{ij}^l \right) \tag{9}$$

$$h_i^{l+1} = \phi_h^l(h_i^l, m_i^l) \tag{10}$$

where $[\cdot, \ldots, \cdot,]$ denotes concatenation. Note that we consider the positions simply as vector-valued features that we embed in a multivector, together with other vectors if available. Here, $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ both denote multivector-valued layers, where the latter one in addition to linear layers also applies (unparameterized) geometric products (Ruhe et al., 2023a) that create higher-order elements in Clifford subspaces, providing the ability to capture complex geometric features. ϕ denote regular MLP layers operating in scalars. These invariant networks are cornerstones of the expressive power and scalability of the Clifford-EGNN.

3.2. Multivector Neurons and MVN-GNN

Vector neurons (VN) typically operate on point cloud data $x \in (\mathbb{R}^n)^{c_{\text{in}}}$, but can be extended to arbitrary vectors v. The core idea is to linearly combine channels of vectors, and apply a vector nonlinearity, completely analogous to regular feed-forward architectures. That is,

$$v_i' = L(v; W)_i = \sum_{j=1}^{c_{\text{in}}} w_{ij} v_j \tag{11}$$

parameterized by weight matrix $W \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$, where w_{ij} is the element in i-th row, j-th column of weight matrix W.

To maintain the expressiveness of the networks other than using only linear combinations of vectors, VN defines an $\mathrm{O}(n)$-equivariant activation function. Given a vector feature $v \in (\mathbb{R}^n)^c$, weight matrix $Q \in \mathbb{R}^{c \times c}$, weight matrix $K \in \mathbb{R}^{c \times c}$, the output v' of activation function is defined as:

$$q = L(v; Q), \qquad k = L(v; K), \tag{12}$$

$$v_i' = \begin{cases} q_i & \text{if } \langle q_i, k_i \rangle \geq 0 \\ q_i - \left\langle q_i, \frac{k_i}{\|k_i\|} \right\rangle \frac{k_i}{\|k_i\|} & \text{otherwise}, \end{cases} \tag{13}$$

where $\langle \cdot, \cdot \rangle$ defines the inner product in Euclidean space. The second case is alwo known as the *rejection* of q from k. Note that there are no scalar inputs in vector neurons, and this hinders the usage of vector neurons in learning on geometric graphs with scalar features.

In Multivector Neurons (MVN), we use similar ideas but instead operate on multivectors. Essentially, we apply linear layers analogous to the ones from Vector Neurons and replace the vector inner product $\langle \cdot, \cdot \rangle$ with its Clifford algebra-extended version $\mathfrak{b}$. That is,

$$\mathbf{v}_i' = \begin{cases} \mathbf{q}_i & \text{if } \mathfrak{b}(\mathbf{q}_i, \mathbf{k}_i) \geq 0 \\ \mathbf{q}_i - \mathfrak{b}\left(\mathbf{q}_i, \frac{\mathbf{k}_i}{\sqrt{\mathfrak{b}(\mathbf{k},\mathbf{k})}}\right) \frac{\mathbf{k}_i}{\sqrt{\mathfrak{b}(\mathbf{k},\mathbf{k})}} & \text{otherwise}\,. \end{cases} \tag{14}$$

Note that this nonlinearity can be extended to more generally use the *rejection* of multivectors (Chisolm, 2012).

As such, a simple two-layer Multivector Neuron MLP (MVN-MLP) then has

$$\mathbf{v}' = \boldsymbol{\phi}_2 \cdot \boldsymbol{\sigma}_1 \cdot \boldsymbol{\phi}_1(\mathbf{v})\,, \tag{15}$$

where $\boldsymbol{\sigma}$ denotes the nonlinearity as introduced in Equation (14) and $\boldsymbol{\phi}$ denotes a multivector linear layer. Additionally, similar to Clifford EGNN, we also introduce scalar networks such that MVN can accept invariant inputs. Formally, with central node index i, neighbor node index j, invariant scalar features $h \in \mathbb{R}^s$, multivector features $\mathbf{v}$, invariant scalar message networks ϕ_e, invariant vector message networks ϕ_v, MVN-MLP, quadratic form $\mathfrak{q}$ in Clifford space, invariant scalar update networks ϕ_h, geometric product layers $\boldsymbol{\psi}_v$, layer index l, we can define the message passing scheme in MVN as follows:

$$\mathbf{v}_{ij}^l = \text{MVN-MLP}^l([\mathbf{v}_i^l, \mathbf{v}_j^l]) \tag{16}$$

$$s_{ij}^l = [h_i^l, h_j^l, \mathfrak{q}(\mathbf{v}_{ij}^l)] \tag{17}$$

$$m_{ij}^l = \phi_e^l(s_{ij}^l) \tag{18}$$

$$m_i^l = \sum_{j \in \mathcal{N}_i} m_{ij}^l \tag{19}$$

$$\mathbf{v}_i^{l+1} = \mathbf{v}_i^l + \frac{1}{\sqrt{|\mathcal{N}_i|}} \boldsymbol{\psi}_v^l \left(\sum_{j \in \mathcal{N}_i} \phi_v^l(m_{ij}^l) \cdot \mathbf{v}_{ij}^l \right) \tag{20}$$

$$h_i^{l+1} = \phi_h^l(h_i^l, m_i^l) \tag{21}$$

Again, we treat node positions x_i as vector features, which we process through their multivector embedding.

3.3. Multivector Perceptrons and MVP-GNN

We again assume a graph where we have scalar node features $h \in \mathbb{R}^s$ but also vector features $v \in (\mathbb{R}^3)^c$. Geometric vector perceptrons (Jing et al., 2021) (GVP) define a layer that operates on both scalars (invariants) and vectors. Invariant inputs $h \in \mathbb{R}^s$ are processed by MLPs ϕ. Vector features $v \in (\mathbb{R}^3)^c$ are processed by weight matrix $W \in \mathbb{R}^c \times \mathbb{R}^c$ acting as a linear combinator of the input vector features. GVP establishes communication between vectors and scalars by using norms of vectors as invariant input to scalar networks. With invariant scalar feature s, vector features v, invariant scalar network ϕ, vector weight matrices W_h, W_μ, sigmoid function σ, a GVP layer is defined as follows:

$$(v_\mu, v_h) = L(v, W_\mu),\ L(v, W_h) \tag{22}$$

$$(s_\mu, s_h) = ||v_\mu||,\ ||v_h|| \tag{23}$$

$$(s', v') = \phi_s(s_h, s),\ \sigma(s_\mu) \cdot v_\mu \tag{24}$$

Then, for a GVP-style message passing layer, one can use GVPs as building blocks for message and update functions. Similar as EGNN, GVP applies MLPs to invariant scalars, scaling and linear combinations to vectors, thus the output scalars are invariant to transformations and output vectors are equivariant to $\mathrm{O}(n)$.

Correspondingly, in Linear Multivector Perceptrons (MVP-Lin), we have an multivector $\mathbf{v}$, multivector feedforward layers $\boldsymbol{\phi}_\mu, \boldsymbol{\phi}_h$, scalar networks ϕ with a feedforward layer defined as follows:

$$(\mathbf{v}_\mu, \mathbf{v}_h) = \boldsymbol{\phi}_\mu(v), \boldsymbol{\phi}_h(v) \tag{25}$$

$$(s_\mu, s_h) = \sqrt{\mathfrak{q}(\mathbf{v}_\mu)}, \sqrt{\mathfrak{q}(\mathbf{v}_h)} \tag{26}$$

$$(s', \mathbf{v}') = \phi(s_h, s),\ \sigma(s_\mu) \cdot \mathbf{v}_\mu \tag{27}$$

To create higher-order subspace elements, with geometric product network $\boldsymbol{\psi}$, multivector feedforward layers $\boldsymbol{\phi}$, scalar networks ϕ, we also define geometric product Multivector Perceptrons (MVP-GP) as follows:

$$\mathbf{w} = \boldsymbol{\psi}(\mathbf{v}) \tag{28}$$

$$\mathbf{v}' = \boldsymbol{\phi}(\mathbf{w} + \mathbf{v}) \tag{29}$$

$$s' = \phi\left(\left[s, \mathbf{v}'^{(0)}\right]\right),\ \mathbf{v}'^{(0)} = s' \tag{30}$$

As such, an MVP-style message passing layer can be defined as:

$$\mathbf{v}_{ij}^l = \boldsymbol{\phi}_e\left([\mathbf{v}_i^l, \mathbf{v}_j^l]\right) \tag{31}$$

$$s_{ij}^l = \phi_e\left([s_i^l, s_j^l]\right) \tag{32}$$

$$(s_{ij}^{l'}, \mathbf{v}_{ij}^{l'}) = (\text{MVP-GP}_e \cdot \text{MVP-Lin}_e)\left(s_{ij}^l, \mathbf{v}_{ij}^l\right) \tag{33}$$

$$(s_i^{l'}, \mathbf{v}_i^{l'}) = \frac{1}{\sqrt{|\mathcal{N}_i|}} \sum_{\mathcal{N}_i} s_{ij}^{l'}, \frac{1}{\sqrt{|\mathcal{N}_i|}} \sum_{\mathcal{N}_i} \mathbf{v}_{ij}^{l'} \tag{34}$$

$$(s_i^{l+1}, \mathbf{v}_i^{l+1}) = (\text{MVP-GP}_v \cdot \text{MVP-Lin}_v)\left(\left[s_i^l, s_i^{l'}\right], \left[\mathbf{v}_i^l, \mathbf{v}_i^{l'}\right]\right) \tag{35}$$

4. Experiments

We apply our models, Clifford EGNNs, MVN-GNN, and MVP-GNN, on two benchmark datasets: the N-Body sim-

Models	MSE (↓)	Seconds/Iteration (↓)	Memory (MiB) (↓)
EGNN (Satorras et al., 2021)	0.0076 ± 0.0003	0.0115 ± 0.0000	334
GVP-GNN (Jing et al., 2021)	0.0069 ± 0.0002	0.0152 ± 0.0001	370
CGENN (Ruhe et al., 2023a)	0.0039 ± 0.0000	0.3240 ± 0.0000	814
WeLNet (Hordan et al., 2024)	0.0036 ± 0.0002	0.0183 ± 0.0001	1322
Clifford EGNN	0.0038 ± 0.0001	0.0211 ± 0.0001	336
MVP-GNN	$\mathbf{0.0035} \pm 0.0000$	0.0431 ± 0.0000	522
MVN-GNN	0.0036 ± 0.0001	0.0274 ± 0.0000	430

Table 1. MSE (↓) of the tested models on the N-body simulation dataset.

Models	MSE (↓)	Seconds/Iteration (↓)	Memory (MiB) (↓)
EGNN (Satorras et al., 2021)	0.392	0.018	6558
GVP-GNN (Jing et al., 2021)	0.678	0.036	11966
CGENN (Ruhe et al., 2023a)	N/A	N/A	OOM
Clifford EGNN	0.244	0.045	8508
MVP-GNN	**0.239**	0.178	24800
MVN-GNN	**0.239**	0.109	20530

Table 2. MSE (↓) of the tested models on the protein backbone structure denoising

ulation dataset (Satorras et al., 2021) and SideChainNet (King & Koes, 2021).

4.1. N-Body System

The N-Body simulation dataset is a popular benchmark for evaluating equivariant deep learning models. The task is to use the initial positions, charges, and velocities of $n = 5$ charged particles to predict their final positions after 1000 time steps. Following Satorras et al. (2021), we simulate 3000 samples for the training set, 2000 samples for the validation set, and 2000 samples for the test set. We use 100 graphs per batch, a learning rate of 5×10^{-3}, and a weight decay of 1×10^{-4} to train our Clifford models. The Mean Square Error (MSE) from our Clifford models and baseline models are presented in Table 1. Our Clifford models outperform all baseline models in terms of MSE. Although our tested Clifford models consume more memory and computational time, they are much more efficient than the Clifford Group Equivariant Neural Networks (CGENN) from Ruhe et al. (2023a). We push the state-of-the-art error on the N-body dataset to 0.0035 (averaged over 3 runs); an 8% improvement over recent methods. These experiments are conducted on an NVIDIA RTX A100 GPU.

4.2. Protein Structure Denoising

SideChainNet (King & Koes, 2021) is a dataset containing all atoms in protein structures for machine learning research. For our experiment, the task is to recover noised 3D protein structures by predicting the positions of each atom. This is inspired by the GitHub user LUCIDRAINS who benchmarks several equivariant architectures using this task (Wang, 2024). For simplicity, we choose to work only with backbone atoms C, N, and O. For a set of backbone atoms $\{C_i, N_i, O_i\}_{i=0}^{n}$, we add Gaussian noise to the coordinates of the backbone atoms. We use the noised backbone atom coordinates as input and use equivariant models to predict the original backbone atom coordinates. For each graph, we connect every node with $k = 16$ nearest neighbors. We use 16 graphs per batch with a learning rate of 1×10^{-3} to train our Clifford models. We choose MSE as our metric to evaluate the models' performance. From Table 2, we observe that Clifford models surpass the baselines by a large margin. Clifford EGNN, in particular, uses even less memory than GVP-GNN for training. These results show that our Clifford models better grasp geometric features while being fast and memory efficient compared to CGENN. These experiments are conducted on an NVIDIA RTX A6000 Ada GPU.

5. Conclusion

In this work, we tested a set of fast and expressive Clifford models: Clifford EGNN, MVP-GNN, and MVN-GNN, based on the design of EGNN, GVP, and VN. These are all equivariant to $\mathrm{O}(n)$. These new parameterizations rely more heavily on fast and expressive scalar networks, that occasionally interact with multivectors that contain higher-order geometric information. We observe promising or even state of the art results on N-Body simulation tasks and protein structure denosing task, while sometimes being more

efficient than the baseline models. Furthermore, this work provides potential directions for the design of Clifford networks that are suitable for large scale graph learning tasks.

Acknowledgments

We would like to thank Maksim Zhdanov for all the helpful discussions.

References

Bekkers, E. J., Vadgama, S., Hesselink, R., der Linden, P. A. V., and Romero, D. W. Fast, Expressive SE(n) Equivariant Networks through Weight-Sharing in Position-Orientation Space. In *The Twelfth International Conference on Learning Representations*, 2024.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and Physical Quantities Improve E(3) Equivariant Message Passing. In *International Conference on Learning Representations*, 2022.

Brandstetter, J., van den Berg, R., Welling, M., and Gupta, J. K. Clifford Neural Layers for PDE Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Brehmer, J., de Haan, P., Behrends, S., and Cohen, T. Geometric Algebra Transformer. In *Neural Information Processing Systems*, 2023.

Chisolm, E. Geometric Algebra, 2012.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. In *International Conference on Computer Vision*, 2021.

Gao, Z., Tan, C., and Li, S. Z. PiFold: Toward Effective and Efficient Protein Inverse Folding. In *The Eleventh International Conference on Learning Representations*, 2023.

Gasteiger, J., Groß, J., and Günnemann, S. Directional Message Passing for Molecular Graphs. In *International Conference on Learning Representations*, 2020.

Hordan, S., Amir, T., and Dym, N. Weisfeiler leman for euclidean equivariant machine learning. In *International Conference on Machine Learning*, 2024.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., and Dror, R. Learning from Protein Structure with Geometric Vector Perceptrons. In *International Conference on Learning Representations*, 2021.

King, J. E. and Koes, D. R. SidechainNet: An All-Atom Protein Structure Dataset for Machine Learning. *Proteins: Structure, Function, and Bioinformatics*, 89(11):1489–1496, Nov 2021. doi: 10.1002/prot.26169. Epub 2021 Jul 12.

Liao, Y.-L. and Smidt, T. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. In *Neural Information Processing Systems*, 2022.

Liu, C., Ruhe, D., Eijkelboom, F., and Forré, P. Clifford Group Equivariant Simplicial Message Passing Networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Ruhe, D., Brandstetter, J., and Forré, P. Clifford Group Equivariant Neural Networks. In *Neural Information Processing Systems*, 2023a.

Ruhe, D., Gupta, J. K., Keninck, S. D., Welling, M., and Brandstetter, J. Geometric Clifford Algebra Networks. In *International Conference on Machine Learning*, 2023b.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) Equivariant Graph Neural Networks. In *International Conference on Machine Learning*, 2021.

Spinner, J., Bresó, V., de Haan, P., Plehn, T., Thaler, J., and Brehmer, J. Lorentz-Equivariant Geometric Algebra Transformers for High-Energy Physics. *arXiv preprint arXiv:2405.14806*, 2024.

Wang, P. Phil Wang's GitHub repositories, 2024. URL https://github.com/lucidrains. Accessed: 2024-05-29.

Zhdanov, M., Ruhe, D., Weiler, M., Lucic, A., Brandstetter, J., and Forré, P. Clifford-Steerable Convolutional Neural Networks. In *International Conference on Machine Learning*, 2024.