Learning Local Equivariant Representations for Catalysis Simulations

Andrea Lombardo
University of Amsterdam
andrealombardo180@gmail.com

Chase van de Geijn
University of Amsterdam
chasevdgeijn@gmail.com

Lasse Becker-Czarnetzki
University of Amsterdam
lasse.becza@gmail.com

Tadija Radusinović
University of Amsterdam
radusinovictadija@gmail.com

Tin Hadži Veljković
University of Amsterdam
tin.hadzi@gmail.com

Abstract

The reliability of renewable energy has been greatly hindered by its inability for large-scale energy storage. One of the most promising approaches is the hydrogen fuel cell which requires an electrocatalyst to be efficient, however, the search space of potential catalysts is vast, making traditional simulations slow. It has been proposed to use deep learning to dramatically speed up this process. Past research has tunneled on message passing graph neural networks as a solution, while recently the Allegro model [1] has shown to achieve state-of-the-art results without relying on message passing. In this paper, we integrate Allegro with the Open-Catalyst dataset (OC20) [2] to investigate its applicability for catalyst generation. To this end, we adapt Allegro for all three tasks in the OC20 dataset and further test Allegro data efficiency.

1 Introduction

Deep neural networks (DNNs) [3] have shown promise in approximating the energies of molecular systems given by Density Functional Theory (DFT) [4] calculations [5] at significantly reduced inference times. This reduction in cost, coupled with DNN's ability to generalize [6], has resulted in hope that a well-trained neural model can be used in situations where conducting extensive DFT simulations would be prohibitively expensive. One such use case is catalyst evaluation [7], due to the large search space of candidate molecules.

This has led to the Open Catalyst Project (OCP), which created a benchmark dataset (OC20) that contains results of hundreds of thousands of DFT simulations, resulting in energies and forces of various configurations of adsorbate molecules at inorganic interfaces [2]. Many deep learning models have been proposed for this dataset [8, 5, 9, 10, 11], which are all variations on the dominant trend of message passing graph neural networks (MPNN)[12]. In MPNNs, the receptive field of layers grows with depth.

The main interaction between atomic particles is the Coulomb force, which falls off quadratically in space. For this reason, many computational chemistry methods assume atoms beyond a certain neighborhood have negligible effect. Thus, there is reason to believe that the growing receptive field of MPNNs is not necessary for describing molecular systems accurately.

With this in mind, Musaelian et al. [1] recently proposed Allegro, a novel graph neural network architecture which is purely localized with a fixed receptive field. This introduces an inductive bias that has shown to increase data efficiency without sacrificing accuracy. Allegro achieves state-of-the-art performance on common benchmarks such as QM9 [13] and MD17 [14].

Submitted to Deep Learning 2 *Mini-project* assignment.

We hypothesize that Allegro will show better data efficiency compared to the results of baseline models on OC20. We compare Allegro, trained on subsets of the whole dataset, to published performances of baseline models.

Our contributions are as follows:

- We benchmark Allegro on all three tasks of the OC20 dataset, investigating its potential for electrocatalyst generation.
- We perform an ablation study on Allegro to investigate the efficacy of restricting interactions to a local neighborhood for electrocatalyst evaluation.
- We modify Allegro to predict vector quantities by proposing two novel approaches: gradient-based translation and magnitude-direction decomposition, as described in Section 4.2.

2 Background

2.1 Open Catalyst 2020

The Open Catalyst[2, 15] dataset is a large dataset of DFT calculations of adsorbate-catalyst systems.[1] The dataset is provided with three main tasks in mind: initial structure to relaxed structure (IS2RS), initial structure to relaxed energy (IS2RE), and structure to energy and forces (S2EF). For training, $640,118$ relaxations are provided across a wide variety of surfaces and adsorbates. The intermediate structures and their corresponding energy and forces are also provided for each relaxation. This results in over 133 million training structures. Four subsplits are used for each of the validation and test sets by considering all combinations of potential extrapolations, as shown in Table 2 in Appendix B.

- **Initial Structure to Relaxed Structure (IS2RS)** - The goal of this task is to take a graph, the initial state of the adsorbate and catalyst, and predict the relaxed, steady state of the system. There are two ways to approach the task: one can iteratively predict the forces and update the state until convergence (similar to DFT), or one can predict the final structure directly. Directly predicting the final state is significantly more computationally efficient, however it has been found to yield less accurate results. As our resources are limited, we will focus on the direct approximation and use MAE as the main metric.
- **Initial Structure to Relaxed Energy (IS2RE)** - The goal of this task is to predict the final, relaxed energy of the system given the initial structure. We report the MAE of the direct prediction the relaxed energy.
- **Structure to Energy and Force (S2EF)** - The goal of this task is to learn to predict the energy and force given a catalyst-adsorbate configuration. Compared with calculating energy and forces using DFT, a neural network prediction is significantly more efficient. This allows for a faster relaxed structure calculation using the same iterative scheme as DFT. In the S2EF task, both the energy and forces are optimized by the loss function. The relative weight of each loss is determined by the *force coefficient*. The main metrics used to evaluate performance are the mean absolute error (MAE) and energy and forces within threshold (EFwT). The EFwT is defined as the percentage of structures with predicted energy within 0.02 eV of the ground truth energy, and with maximum error in forces below 0.03 eV$/$Å. Both of these criteria must be met for the structure to be labeled as "correct".

3 Related work

3.1 Baselines

OC20 proposes three baseline models for benchmarking: SchNet [8], which applies convolutional neural networks to quantum chemistry by introducing a continuous convolution kernel. Crystal Graph Convolutional Neural Network (CGCNN) [5] represents molecular systems as a graph with its bonds as edges. Finally, the best performing benchmark is **Di**rectional **Me**ssage-passing Neural Network (DimeNet) [9] which introduces directional edges to the graph. Allegro, like DimeNet, uses directed edge embeddings achieved by projecting the direction of edges to the spherical harmonics basis. By contrast, Allegro uses more advanced tensor product layers in addition to the key property of limiting

[1] The dataset also includes molecular dynamics data, relaxation trajectories and other supplementary components, which are not of interest for this project.

the receptive field. We explore the baselines in more detail in Appendix C. Allegro and all of the baselines account for energy invariance to translations and rotations in some manner.

3.2 Equivariant Networks

Molecular geometry plays an essential role in chemical and physical properties of all molecules and polymers [16, 17]. For example, conformation geometry — the spatial arrangement of residues in a protein — will govern its function [18]. Similarly, chemical properties of molecules and crystals also heavily depend on the geometry of their components [19]. Enforcing a corresponding geometric constraint in a model will result in a reduction of the problem dimensionality (reducing the model variance) while not introducing a bias [20]. For this reason, including geometric priors in graph neural networks has been providing strong foundations for current state-of-the-art molecular graph approximations [21]. In modern equivariant networks, layers are constructed using tensor products, which naturally handle geometric quantities such as orientations [1, 22, 23]. In Appendix A, we explain how modern networks like Allegro implement equivariant layers using aforementioned tensor products.

Currently, the state-of-the-art for all tasks in the OC20 dataset is GemNet [10], which is a refined DimeNet with two-hop message passing that respects geometric symmetries.

3.3 Message Passing

All four models we have discussed so far have been variants of graph neural network (GNN) and, more specifically, message passing graph neural networks (MPNNs) [24, 25, 12]. In GNNs, a node passes its information to neighboring nodes. What makes these models message passing is that this signal will be propagated further by its neighbors with each layer in the network, progressively increasing its receptive field.

We define the *receptive field* as the set of nodes in the graph that are affected by the message of the origin node. In each iteration of the graph, the receptive field grows. We define the number of *hops* between A and B as the number of edges needed to traverse the graph from node A to node B. The receptive field of a node after N layers of an MPNN will be the set of nodes within N hops of the message origin.

Message passing has continued to be a backbone of all state-of-the-art neural networks for learning interatomic potentials. Success of MPNNs has been attributed to this ability of distant nodes to communicate. However, recent work has shown that using localized messages is sufficient in some tasks.

4 Approach

Our goal is to benchmark the Allegro model on the OC20 dataset, further probing its supposed data-efficiency on a dataset significantly larger than what has been done in the original paper. Since the source code of Allegro[2] is based on NequIP [22], we needed to modify its data processing methods to integrate it with the OC20 training pipeline. In addition, we implemented Weights & Biases [26] (W&B) logging support for the OC20 training in order to track experiments more easily. We focus on its performance on predicting energies and forces of a given configuration. Finally, we experiment with extending Allegro to directly accommodate relaxed structure prediction from an initial structure, bypassing the iterative scheme of structure optimization. Our fork of the OCP repository is publicly available. [3]

4.1 Allegro

Allegro is by construction an E(3) equivariant neural network. The E(3) group is the 3D Euclidean group which consists of translations, rotations and reflections. The energy of our system should be invariant to these transformations, while the forces and relaxed structure should be equivariant. An invariant output can be achieved by using an invariant aggregator, in this case a summation. Allegro, as proposed in the Musaelian et al., takes in the molecular graph as input, and outputs the E(3)

[2] https://github.com/mir-group/allegro

[3] https://github.com/rtadijar/ocp

invariant energy as

$$E_{\text{system}} = \sum_{i}^{N} E_i, \tag{1}$$

where N is the number of nodes in the graph, and E_i is the total energy exerted at node i. However, information in the graph is not stored as node-level features. Rather, energy is calculated from edge-level features,

$$E_i = \sum_{j \in \mathcal{N}(i)} E_{ij}, \tag{2}$$

where E_{ij} are the output features of the ij edge. Predicting energies as edge-level features is one of the key differences compared to other MPNNs. It is important to note that $E_{ij} \neq E_{ji}$, as this allows for both directional messages and a constant receptive field in the subsequent layers. Finally, the force on the i-th atom can be calculated as a negative gradient of the energy with respect to the position of the i-th atom:

$$\vec{F_i} = -\vec{\nabla}_i E_{\text{system}}. \tag{3}$$

Calculating forces in this manner ensures that the forces give an energy-conserving field, which is especially important for longer molecular dynamics simulations.

On the right of the Allegro schematic in Figure 1, the scalar and the tensor track interact in multiple ways. First, the weights in the main tensor product are calculated using the scalar-level features. Second, following the tensor product, the scalar features of the new tensor are concatenated with the scalar track, yielding updated scalar-level features. Allegro only uses features from the scalar track for calculating the energy. More details on the Allegro architecture can be found in Appendix D

This asymmetric, edge-level approach allows for parallelization across multiple workers, as the iterative propagation of MPNNs is absent.

4.2 Allegro IS2RS task

As we have discussed, the output of the Allegro model is the final feature-vector of the scalar ($l = 0$) track. This works for predicting scalar quantities such as energies, however, it does not work for geometric quantities such as positions. Therefore, for the IS2RS task, we must modify the Allegros architecture.

Let $\mathbf{r}_i$ denote the set of initial node positions, and $\mathbf{r}_f$ the set of relaxed structure positions, both of which have the shape $3 \times N$ for N number of nodes. To ensure translational equivariance of our model, we predict the change in position $\mathbf{r}_d \equiv \mathbf{r}_f - \mathbf{r}_i$, rather than directly predicting the final positions $\mathbf{r}_f$. We propose two novel equivariant approaches:

Gradient-based translation

In this approach, we make minimal adjustments to the model. We propose a similar approach to the one used for force predictions, as physically, forces will govern the motion of the molecule. For this reason, we expect reasonable performance as the two tasks are similar.

In this approach, we calculate the final node-level features from the scalar track as follows:

$$\tilde{r} = \sum_{i}^{N} \sum_{j \in \mathcal{N}(i)} \tilde{r}_{ij}, \tag{4}$$

where we perform sums in the same way as for the energy predictions. The quantity $\tilde{r}$ is a scalar-valued feature, from which the translation difference vector is obtained for node i by taking the gradient with respect to its coordinates:

$$\mathbf{r}_{d_i} = -\vec{\nabla}_i \, \tilde{r} \tag{5}$$

Obtaining a vector field by taking the gradient of a scalar function ensures that the field is conservative. In the case of energy prediction, this assumption was a physical prior, since the Coulomb's force is a conservative force. There is, however, no immediate interpretation of a conservative field for translation vectors. In the following sections, we discuss whether this imposed constraint is harmful for the performance of the model.

Table 1: Performance of baseline models and the Allegro models on the validation set for different subsets of the training set. In this notation, l_k denotes $l_{\text{max}} = k$.

	S2EF				IS2RE			IS2RS
	200K		2M		10K	100K	Full	100k
	Force MAE	EFwT	Force MAE	EFwT	Energy MAE			Pos. MAE
CGCNN	0.0800	0.00%	0.0673	0.01%	0.9881	0.6820	0.6199	–
SchNet	0.0743	0.00%	0.0737	0.00%	1.0590	0.7137	0.6458	0.1370
DimeNet	**0.0693**	**0.01%**	**0.0576**	**0.02%**	1.0117	**0.6658**	**0.5999**	–
Allegro (l_1)	0.0781	0.00%	**TBA**	**TBA**	**0.9377**	0.7463	0.6863	–
Allegro (l_2)	–	–	–	–	0.9385	0.7479	0.6781	**0.1350**
Allegro (l_3)	–	–	–	–	0.9398	0.7582	0.7082	–

Magnitude-direction decomposition

In this approach utilizes both the scalar and the tensor track to obtain the translation difference vector in the following decomposition:

$$\mathbf{r}_{d_i} = |\mathbf{r}_{d_i}| \, \hat{\mathbf{r}}_{d_i} \tag{6}$$

We use the scalar track of Allegro to output the magnitude in of the vector, which is calculated as described in equation 4. The directional unit vector $\hat{\mathbf{r}}_{d_i}$ is calculated from the tensor-track features at the final layer $L = k$ as:

$$\hat{\mathbf{r}}_{d_i} = \frac{\sum_{j \in \mathcal{N}(i)} \sum_n \mathbf{V}^{ij,L=k}_{n,l=1,m}}{\| \sum_{j \in \mathcal{N}(i)} \sum_n \mathbf{V}^{ij,L=k}_{n,l=1,m} \|_2}, \tag{7}$$

where we sum only over the features whose $l_{\text{max}} = 1$, as the translation difference vector transforms in the same way as the tensors of $l = 1$. This operation can also be seen as a local averaging the output tensors whose $l = 1$. Unfortunately, we were unable to implement and test this proposed algorithm due to the time constraints of this project.

5 Experiments

Here, we investigate Allegro performance on OC20 by benchmarking its performance on the S2EF and IS2RE task. Furthermore, we test the authors' claim [1] that Allegro is data efficient by comparing results on the smaller subsets of the IS2RE dataset. In addition, we perform an ablation study on the effect of l_{max} which controls the band limit imposed on the output resulting from tensor products in the Allegro layers. Setting this to 3, as done by the authors, results in a significantly higher computational cost. Reducing it therefore speeds up training, while hindering performance. We use all the same hyperparameter settings as the authors [1] with 3 Allegro layers. We perform a W&B hyperparameter sweep. We put 0.002 as the initial learning rate and set the force coefficient to 1000 for the S2EF task. For the sake of reproducibility we provide a full training setup in the Appendix E.

5.1 Results

In Table 1, we compare Allegro to the baseline performance trained on different train data subsets with the most relevant metrics. Results are compared on the in-domain validation set since it's the only one with reported values for different training subset splits. [4]

All Allegro trainings on the S2EF task resulted in very poor performance for the computed forces. After thorough investigation a bug was found in the backward pass of Allegro. The base Allegro implementation did not calculate the gradients for forces, only energy, and thus couldn't learn properly. Due to limited resources and time, we were only able to train a simple 1 layer Allegro model on the 200K dataset subset. While we can't analyse Allegro true performance on that task we at least see a proof of concept for it's applicability as it is competitive with the baselines. It is also noticeable that even this small model hasn't converged in the limit time we were able to train it (see 3).

[4] The performance of baselines on the in-domain validation set is given in https://github.com/Open-Catalyst-Project/ocp/blob/main/MODELS.md

5.2 Ablation Study

We investigate the trade-off between the computational speed benefits and improved accuracy for different values of l_{max} on the IS2RE task. We observe that a higher l_{max} does not lead to better performance, which we ascribe to the model not being well suited for the task. We discuss this further in Section 5.2.3.

5.2.1 Effect of Training Data

We investigate the Musaelian et al.'s 1 claim that Allegro is data efficient by comparing results when training on different subsets (see Table 1). We can observe that for the IS2RE task, Allegro (l_1) model outperforms other baselines when trained on the smallest subset (10K). However, the relative increase of performance when trained on larger subsets (100K and Full) is smaller for Allegro compared to other baselines. This indicates that Allegro is more data efficient than the baselines, however, the inferior performance on larger datasets presumably originates from the fact that Allegro locality is not suited for this task resulting in a higher bias. We also observe that Allegro shows a very drastic initial improvement within the first few training steps before plateauing to a more gradual increase in performance (see Figure 2).

5.2.2 Allegro IS2RS

Here, we compare the results of our IS2RS Allegro method with baseline methods. Note that, due to time constraints, our results only include the SchNet baseline and one Allegro implementation, as the direct computation of the IS2RS was not implemented, nor reported. We implement only the Gradient-based translation, as the Magnitude-direction decomposition required severe modifications of the Allegro code, which we leave for future work. We can observe that both networks converge to similar values, both of which underperform compared to the iterative methods. This is expected, as a direct predictions are a much harder task.

5.2.3 Allegro Limitations

Although Allegro achieved SOTA results on many datasets, we have observed that it barely outperformed baseline models on the OC20 tasks. We hypothesize that this is due to the fact that the OC20 dataset is inherently different than the previous datasets. Datasets like QM9 [13] consist of isolated molecules, and the predicted properties are of the molecule itself. On the other hand, OC20 [2] data consists of a large flat surfaces (the catalyst) and adsorbate molecules, and the tasks involve predicting properties of this dual system. While Allegro localized interactions are physically-inspired, there is a caveat with this argument for the catalyst systems. When an adsorbate is near the surface, the catalyst atoms which are further away have minor contributions individually, but since there are many surface atoms, the collective contribution is no longer negligible and Allegro cannot account for this. We believe this problem only occurs when dealing with systems of asymmetric sizes, such as this one (large surface and a small adsorbate molecule). For this reason, we assume that message passing outperforms localized networks, as the long-range interactions between distant nodes are important.

This effect is also more apparent in the direct tasks, such as IS2RE and IS2RS. We believe this is due to the fact that in these tasks, the model has to directly predict the properties of the system in the relaxed state. Since the input graph is the initial state, the adsorbate will not be able to interact with deeper layers of the catalyst. For the same reason we believe that this effect is less apparent in the S2EF task, as there we predict the properties of the system in the current configuration, rather than the properties of the system in a distant relaxed state.

6 Conclusion

In this work, we adapt Allegro to be trained on the OC20 dataset, investigating its applicability for catalyst generation and discovering potential shortcomings. We found that the inductive bias of localized atomic contribution allows for more efficient learning on small datasets, but hinders the data scaling performance on the IS2RE task due to the importance of global information. We also indicate Allegro viability for the S2EF task with more in-depths studies being left for future work. Finally, we propose two novel methods to directly predict the IS2RS structures: gradient-based, yielding unsatisfactory results; translation and magnitude-direction decomposition, which is left for future work.

References

[1] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics, 2022.

[2] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, et al. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 11(10):6059–6072, 2021.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] Nathan Argaman and Guy Makov. Density functional theory: An introduction. *American Journal of Physics*, 68(1):69–79, jan 2000.

[5] Cheol Woo Park and Chris Wolverton. Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery. *Physical Review Materials*, 4(6), jun 2020.

[6] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.

[7] Sebastian Matera, William F. Schneider, Andreas Heyden, and Aditya Savara. Progress in accurate chemical kinetic modeling, simulations, and parameter estimation for heterogeneous catalysis. *ACS Catalysis*, 9, 6 2019.

[8] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. 2017.

[9] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs, 2020.

[10] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules, 2021.

[11] Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik Bekkers, and Max Welling. Geometric and physical quantities improve E(3) equivariant message passing. *CoRR*, abs/2110.02905, 2021.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Message passing neural networks. In *Machine learning meets quantum physics*, pages 199–214. Springer, 2020.

[13] Raghunathan Ramakrishnan, Pavlo Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules, Jul 2014.

[14] Anders S Christensen and O Anatole von Lilienfeld. On the role of gradients for machine learning of molecular energies and forces. *Machine Learning: Science and Technology*, 1(4):045018, 2020.

[15] C. Lawrence Zitnick, Lowik Chanussot, Abhishek Das, Siddharth Goyal, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Thibaut Lavril, Aini Palizhati, Morgane Riviere, Muhammed Shuaibi, Anuroop Sriram, Kevin Tran, Brandon Wood, Junwoong Yoon, Devi Parikh, and Zachary Ulissi. An introduction to electrocatalyst design using machine learning for renewable energy storage, 2020.

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[18] Derek HR Barton. The principles of conformational analysis. *Science*, 169(3945):539–544, 1970.

[19] Hugh PG Thompson and Graeme M Day. Which conformations make stable crystal structures? mapping crystalline molecular geometries to the conformational energy landscape. *Chemical Science*, 5(8):3173–3182, 2014.

[20] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[21] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *CoRR*, abs/2102.09844, 2021.

[22] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1), may 2022.

[23] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

[24] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[25] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[26] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[27] Real-time operating grid - u.s. energy information administration (eia).

[28] Francisco Gonzalez Ledesma and Matthew Mewes. Spherical-harmonic tensors. *Physical Review Research*, 2(4), oct 2020.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[31] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019.

[32] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016.

[33] Public leaderboard of open-catalyst project. `https://opencatalystproject.org/leaderboard.html`.

[34] Miltiadis Kofinas, Naveen Nagaraja, and Efstratios Gavves. Roto-translated local coordinate frames for interacting dynamical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

[35] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

Appendix

A Equivariance and Spherical Harmonics

Introducing geometric quantities, such as positions or directions, requires defining their local frame [34]. Since certain scalar quantities — such as energy and heat capacity — are *invariant* to the position and/or orientation of the local coordinate frame. Therefore, including geometric quantities in the networks has to be designed carefully to satisfy these constraints. On the other hand, vector quantities (such as forces and dipole moments) are *equivariant* to the rotations of the local coordinate frame.

Currently, all state of the art neural networks in this domain utilize the equivariance with respect to the E(3). In EGNN [21], authors propose a message passing scheme in which the interatomic distance is appended as an edge-wise feature. However, with this approach, the directional information is not fully utilized, as the interatomic distances are completely *invariant* features. For this reason, in the SEGNN [11], authors include directional information by projecting the direction of edges to the spherical harmonics basis.

Spherical harmonics are defined as:

$$Y_{lm}(\theta, \varphi) = N e^{im\varphi} P_l^m(\cos\theta), \tag{8}$$

where θ and φ are spherical angles, N is a normalization constant, and $P_l^m(\cos\theta)$ is the associated Legendre polynomial. The Legendre polynomials are determined by two indices: l and m. Firstly, l is a non-negative integer, while, for a given l, m is an integer in the range $m \in [-l, \ldots, l]$. Crucially, spherical harmonics serve as a complete basis for functions on the sphere. Therefore, directional unit-vectors, which can be seen as a Dirac δ-function on the sphere, can be decomposed in this basis. Carefully utilizing coefficients of such decomposition (up to some order l_{max}), along with the use of tensor products [28] instead of regular matrix-vector multiplication serves as a backbone of many equivariant graph neural networks. A more detailed summary of the math involved in the construction of neural networks that utilize tensor products can be found in the appendix of [11].

B OC20 Dataset Splits

Table 2: Size of test/validation splits (number of structures for S2EF and initial structures for IS2RS and IS2RE). The subsplits include In-Domain (sampled from the training distribution), Out-of-Domain Adsorbate (OOD Adsorbate), OOD Catalyst, and OOD Both (both unseen adsorbate and unseen catalyst compositions).

Task	Train	In-Domain	OOD Adsorbate	OOD Catalyst	OOD Both
S2EF	133,953,162	1,000,000	1,000,000	1,000,000	1,000,000
IS2RS	460,364	24,946	24,966	24,963	24,988
IS2RE	460,364	24,946	24,966	24,963	24,988

C OC20 Baselines

In this section we present baselines used to benchmark the OC20 dataset in more details.

SchNet

SchNet [8] was the earliest of the benchmarks, predating much of the recent research in geometric deep learning. In computer vision, convolutions had found a lot of success. However, in vision, both the kernel and image are discrete grids, whereas in computational physics particles can lie anywhere in the continuous plane. To adapt to new domain, they introduced the continuous convolution kernel,

$$\mathbf{x}_i^{l+1} = \sum_j \mathbf{x}_j^l \circ \mathbf{W}^l(\mathbf{r}_i - \mathbf{r}_j), \tag{9}$$

where $\mathbf{r}_i$ is the 3-dimensional location of particle i, $\mathbf{x}$ is an n-dimensional feature vector and l is the layer number. $\mathbf{W}$ is a learned kernel function that maps $\mathbb{R}^3 \to \mathbb{R}^n$. This innovation lets the kernel be evaluated in a continuous spatial domain and makes use of physical distance between particles.

Crystal Graph Convolutional Neural Networks

The main contribution of this paper [5] was the interpretation of the system as a graph. The particles can be interpreted as the nodes of the graph and the bonds as the edges. CGCNN formed a graph with multiple edges between two particles which represented the different bonds of the lattice.

SchNet can be reinterpretted to fit this paradigm as well. The connections would then be represented as the distance between the atoms rather than the bonds. As SchNet does not take the lattice into account, the graph is fully connected with symmetric edges, meaning there are $\frac{1}{2}N^2$ edges, where N is the number of particles in the system.

DimeNet

The previous methods relied primarily on $\|\mathbf{r}_i - \mathbf{r}_j\|_2$, which discards information about the direction between the atoms. DimeNet [9] introduced **di**rectional **me**ssages. In addition, DimeNet incorporated spherical Bessel basis functions for edges, rather than the Gaussian radial functions of SchNet.

D Allegro Architecture

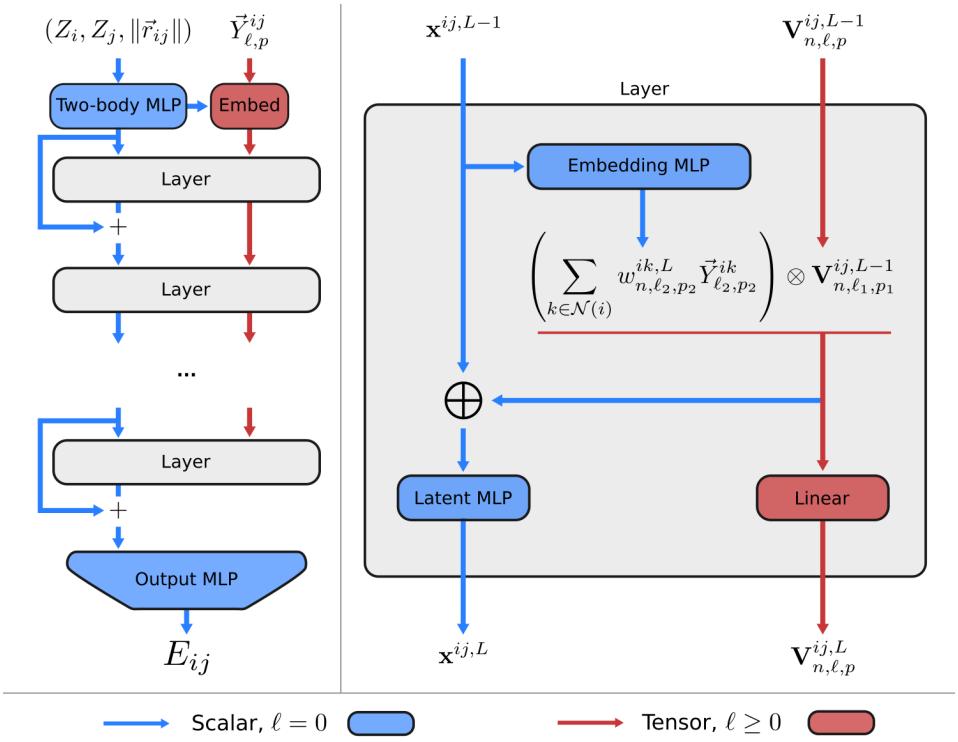

Figure 1: Architecture of the Allegro network. Left: the full Allegro model architecture, with outlined scalar track (blue) and tensor track (red). Right: overview of the main Allegro layer. Scalar-track features are used to determine the weights for the tensor product. The scalar ($l = 0$) features of the updated tensor-track are concatenated and passed through the Latent MLP to update the scalar-track features, allowing for the two tracks to interact. Figure from [1].

The Allegro architecture is shown in Fig 1. On the left we can see the global Allegro architecture, whose inputs are pairs of features ij. The initial *scalar* module, called *Two-body MLP*, takes as input one-hot encodings of atom types of nodes i and j, along with their mutual distance embedded onto the Bessel basis function and passes their concatenated version through a linear MLP. More formally, this reads:

$$\mathbf{x}^{i,j,L=0} = \text{MLP}\left(\text{1HOT}(Z_i) \,||\, \text{1HOT}(Z_j) \,||\, B(r_{ij})\right), \tag{10}$$

where $\text{1HOT}(Z_i)$ is the one-hot encoding of the atom type of node i, $||$ denotes concatenation, and $B(r_{ij})$ is the projection of the distance between nodes i and j onto the Bessel basis.

On the other hand, the initial *equivariant* tensor-track features are computed as a linear embedding of the spherical harmonic projection of the directional unit-vector $\hat{\mathbf{r}}_{\mathbf{ij}}$ (denoted as $\vec{Y}^{ij}_{l,p}$ in Fig. 1). The

unit vector $\hat{\mathbf{r}}_{\mathbf{ij}}$ points from the node i to the node j. These features can be expressed as:

$$\mathbf{V}_{n,l,p}^{ij,L=0} = w_{n,l,p}^{i,j,L=0} \vec{Y}_{l,p}^{ij}, \tag{11}$$

where the weights $w_{n,l,p}^{i,j,L=0}$ are calculated by passing the initial scalar track features through another MLP. More detailed description of these layers can be found in [1].

Unlike usual MPNNs, Allegro operates on *pairs* of features (denoted by ij), in contrast to the usual node-level features. Throughout the network, the information flows through two distinct tracks: a scalar and a tensor track. Scalar features ($l = 0$) are fully invariant to actions of the E(3) group, while the tensor features ($l > 0$) are equivariant by construction. In the features of the tensor-track, L denotes the layer number, n denotes the channel dimension, l is the azimuthal number, while p is the spherical harmonic parity. Spherical number m is implicitly assumed to be present in all tensor-related terms and calculations, but is omitted for notational simplicity.

Although the network outputs only the scalar features, the scalar features interact with the tensor features throughout the network. Specifically, in the *Embedding MLP* layer, the weights of the tensor product are conditioned on the scalar track features, and in this way the scalar features affect the output tensor features. After the tensor product, the scalar part of the output tensor (the $l = 0$ component) is concatenated with the current scalar features and they are passed through the *Latent MLP* layer, allowing the tensor track to affect the scalar features. These rich interactions between scalar and tensor features allow the scalar outputs to benefit from the geometric information passed throughout the network.

As mentioned before, Allegro has a constant receptive field which doesn't grow with subsequent message layers. The locality of the Allegro model emerges from the fact that edge-wise features V^{ij} (other feature indices are omitted for simplicity) are asymmetric, i.e. $\mathrm{V}^{ij} \neq \mathrm{V}^{ji}$. This means that when the j-th node is updated throughout the network, it will not have access to the features of the i-th node, V^{ij}. Therefore, the j-th node will not be able to access the features of the i-th nodes' neighbor k, unless the k-th node is also neighbor of node j. For this reason, even after several Allegro layers, the receptive field remains constant and localized.

E Training

E.1 Setup

For all tasks we set the initial learning rate to 0.002. The model was trained with the `Adam` optimizer with default parameter of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ with no weight decay [29, 30, 31]. We used a `ReduceLROnPlateau` scheduler with a patience of 5, reducing the learning rate by a factor of 0.8 and validating every epoch. Using smaller l_{max} frees up GPU memory, so we increased the batch size for training accordingly (i.e. for $l_{\text{max}} = 1$ we used 4 as batch size for $l_{\text{max}} = 2$ we used 16 and for $l_{\text{max}} = 1$ the batch size was equal to 32). We trained on a single NVIDIA GeForce 1080Ti GPU. We use 3 Allegro layers, a 6.0Å radial cutoff, a trainable Bessel basis of size 8 for the basis encoding with a polynomial envelope function using $p = 6$. The *Two-body MLP* consists of 4 hidden layers of dimensions $[64, 128, 256, 512]$, using a uniform initialization and `SiLU` nonlinearities on the outputs of the hidden layers [32]. The Latent MLP consists of one single layer of dimension $[512]$ and uses the same initialization and nonlinearities as the *Two-body MLP*. The final MLP (called *Output MLP* in Figure 1) has one single layer of dimension $[128]$ with a uniform initialization.

E.2 Training plots

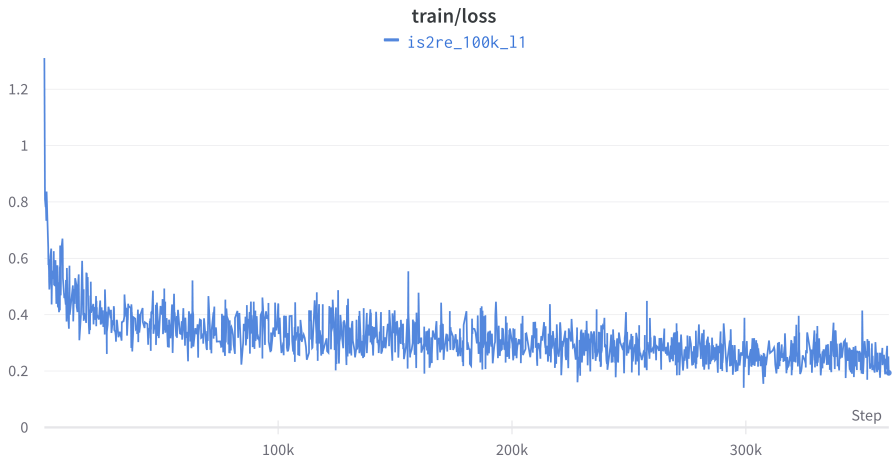

Figure 2: This shows the training loss progression of Allegro trained on the IS2RE task with $l_{\text{max}} = 1$.

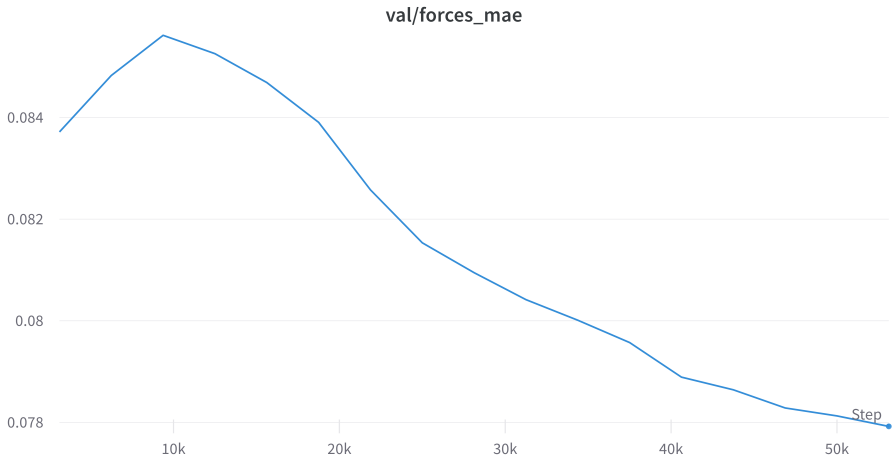

Figure 3: This shows the Energy MAE on the in-domain validation set during training of Allegro on the 200k S2EF dataset.