GemNet: Universal Directional Graph Neural Networks for Molecules

Johannes Klicpera, Florian Becker, Stephan Günnemann
Technical University of Munich, Germany
{klicpera,beckerf,guennemann}@in.tum.de

Abstract

Effectively predicting molecular interactions has the potential to accelerate molecular dynamics by multiple orders of magnitude and thus revolutionize chemical simulations. Graph neural networks (GNNs) have recently shown great successes for this task, overtaking classical methods based on fixed molecular kernels. However, they still appear very limited from a theoretical perspective, since regular GNNs cannot distinguish certain types of graphs. In this work we close this gap between theory and practice. We show that GNNs with directed edge embeddings and two-hop message passing are indeed universal approximators for predictions that are invariant to translation, and equivariant to permutation and rotation. We then leverage these insights and multiple structural improvements to propose the geometric message passing neural network (GemNet). We demonstrate the benefits of the proposed changes in multiple ablation studies. GemNet outperforms previous models on the COLL, MD17, and OC20 datasets by 34 %, 41 %, and 20 %, respectively, and performs especially well on the most challenging molecules. Our implementation is available online. [1]

1 Introduction

Graph neural networks (GNNs) have shown great promise for predicting the energy and other quantum mechanical properties of molecules. They can predict these properties orders of magnitudes faster than methods from quantum chemistry – at comparable accuracy. GNNs can thus enable the accurate simulation of systems that are orders of magnitude larger. However, they still exhibit severe theoretical and practical limitations. Regular GNNs are only as powerful as the 1-Weisfeiler Lehman test of isomorphism and thus cannot distinguish between certain molecules [45, 60]. Moreover, they require a large number of training samples to achieve good accuracy.

In this work we first resolve the questionable expressiveness of GNNs by proving sufficient conditions for universality in the case of invariance to translations and rotations and equivariance to permutations; and then extending this result to rotationally equivariant predictions. Simply using the full geometric information (e.g. all pairwise atomic distances) in a layer does not ensure universal approximation. For example, if our model uses a rotationally invariant layer we lose the relative information between components. Such a model thus cannot distinguish between features that are rotated differently. This issue is commonly known as the "Picasso problem": An image model with rotationally invariant layers cannot detect whether a person's eyes are rotated correctly. Instead, we need a model that preserves relative rotational information and is only invariant to *global* rotations. To prove universality in the rotationally invariant case we extend a recent universality result based on point cloud models that use representations of the rotation group $\mathrm{SO}(3)$ [18]. We prove that spherical representations are actually sufficient; full $\mathrm{SO}(3)$ representations are not necessary. We then generalize this to rotationally equivariant predictions by leveraging a recent result on extending invariant to equivariant

[1] https://www.daml.in.tum.de/gemnet

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

predictions [57]. We then discretize spherical representations by selecting points on the sphere based on the directions to neighboring atoms. We can connect this model to GNNs by interpreting these directions as directed edge embeddings. For example, the embedding direction of atom a would be defined by atom c, resulting in the edge embedding e_{ca}. Updating the spherical representation of atom a based on atom b then corresponds to two-hop message passing between the edges e_{ca} and e_{db} via e_{ba}, with atoms c and d defining the embedding directions. This message passing formalism naturally allows us to obtain the molecule's full geometrical information (distances, angles, and dihedral angles), and the direct correspondence proves the model's universality.

We call this edge-based two-hop message passing scheme *geometric message passing*, and propose multiple structural enhancements to improve the practical performance of this formalism. Based on these changes we develop the highly accurate and sample-efficient geometric message passing neural network (GemNet). We furthermore show that stabilizing the variance of GemNet's activations with predetermined scaling factors yields significant improvements over regular normalization layers.

We investigate the proposed improvements in a range of ablation studies, and show that each of them significantly reduces the model error. These changes introduce little to no computational overhead over two-hop message passing. Altogether, GemNet outperforms previous models for force predictions on COLL by 34 %, on MD17 by 41 %, and on OC20 by 20 % on average. We observe the largest improvements for the most challenging molecules, which exhibit dynamic, non-planar geometries. In summary, our contributions are:

- Showing the **universality** of spherical representations and two-hop message passing with directed edge embeddings for rotationally equivariant predictions.
- **Geometric message passing**: Symmetric message passing enhanced by geometric information.
- Incorporating all proposed improvements in the **Geometric Message Passing Neural Network (GemNet)**, which significantly outperforms previous methods for molecular dynamics prediction.

2 Related work

Machine learning potentials. Research on predicting a molecule's energy and forces (so-called machine learning potentials) started with hand-fitted analytical functions and then gradually moved towards fully learned models. Arguably, classical force fields are their very first instances. They use analytical functions with coefficients that were hand-tuned based on experimental data. A popular example for these is the Merck Molecular Force Field (MMFF94) [29]. The next wave of methods used kernel ridge regression based on fixed, hand-crafted molecular representations [3, 9, 23]. Finally, modern research mostly focusses on fully end-to-end learnable models based on GNNs [28, 53]. These models can also be combined with molecular features from quantum mechanical calculations to improve performance [48]. We consider this combination as orthogonal research.

Directional GNNs. We can also achieve equivariance and invariance to rotations without relying on group representations. Directional GNNs achieve this by representing directional information explicitly [54] or in the form of angles [35] and dihedral angles [25, 39]. Our work is focused on this class of models, proving their universality and proposing an improved variant, GemNet.

Expressiveness of GNNs. A large part of GNN research has been focused on their (limited) expressiveness. Morris et al. [45], Xu et al. [60] first proved that they are only as expressive as the Weisfeiler-Lehman test of isomorphism and Garg et al. [27] showed the limitations of basic directional message passing. Kondor et al. [37], Maron et al. [43], Morris et al. [45, 46] then investigated higher-order representations to circumvent this issue. Finally, Azizian & Lelarge [2], Maron et al. [42] showed that so-called folklore GNNs are the most expressive GNNs for a given tensor order.

Equivariant neural networks. Equivariance and invariance have recently emerged as one of the foundational principles of modern neural networks [13, 15]. This is especially relevant for models in physics, for which we often know the symmetries a priori. Equivariant models for the $\mathrm{SO}(3)$ group were first investigated in the context of spherical convolutions by Cohen et al. [14], Esteves et al. [21], Kondor et al. [36]. These methods leverage group representations to achieve full equivariance. They were then transferred to the context of 3D point clouds and molecules by Anderson et al. [1], Thomas et al. [56], Weiler et al. [58], and further developed by Batzner et al. [4], Finzi et al. [24], Fuchs et al. [26]. Importantly, Yarotsky [61] proved the universality of 2D convolutional networks, and Bogatskiy et al. [5] extended this result to general groups. Maron et al. [44] proved

universality for models invariant to S_n and equivariant to an additional symmetry. Dym & Maron [18] combined these results to prove universality for the joined group of translations, rotations, and permutations. Apart from reflections this is the exact group relevant for general molecules.

3 Universality of spherical representations

GNNs for molecules typically incorporate directional information in one of two ways: Via $\mathrm{SO}(3)$ representations [1, 56] or by using directions in real space [35, 54]. Directions in real space are associated with the three-dimensional S^2 sphere, while the $\mathrm{SO}(3)$ group is double covered by the four-dimensional S^3 sphere. Directional representations thus use one degree of freedom less than $\mathrm{SO}(3)$ representations, making them significantly cheaper. And, as we will prove in this section, directional representations actually provide the same expressivity as $\mathrm{SO}(3)$ representations for predictions in $\mathbb{R}^3$. We achieve this by showing that the $\mathrm{SO}(3)$-based tensor field network (TFN) [56] variant used by Dym & Maron [18] is equivalent to a similar model based on spherical representations, in the case of rotationally invariant predictions. We then generalize a recent result by Villar et al. [57], which lets us extend our theorem to the rotationally equivariant case. Afterwards, we relate this universality to directional GNNs by interpreting them as a discretization of spherical representations.

Preliminaries. We consider a point cloud with n points (atoms), each associated with a position and a set of rotationally invariant features (e.g. atom types), defined as $\boldsymbol{X} \in \mathbb{R}^{3\times n}$ and $\boldsymbol{H}_{\text{in}} \in \mathbb{R}^{h\times n}$. In this section we define model classes by sets of functions $\mathcal{F}$. As a first step, we are interested in proving that the set $\mathcal{F}$ defining our model is equal to the full set of functions $\mathcal{G}'$ that are invariant to the group of translations $\mathbb{T}^3$ and rotations $\mathrm{SO}(3)$, and equivariant to the group of permutations S_n. We denote the codomain of functions in $\mathcal{G}'$ as W_{T}^n, where W_{T} is some representation of $\mathrm{SO}(3)$. We denote a vector's norm by $x = \|\boldsymbol{x}\|_2$, its direction by $\hat{\boldsymbol{x}} = \boldsymbol{x}/x$, and the relative position by $\boldsymbol{x}_{ba} = \boldsymbol{x}_b - \boldsymbol{x}_a$. Proofs are deferred to the appendix. Note that this section is not intended as an introduction to the $\mathrm{SO}(3)$ group. For a concise introduction in the context of machine learning see e.g. Weiler et al. [58, Section 3] or Kondor et al. [36].

Tensor field network. In order to show the equivalence of the TFN to spherical representations, we first need to define this model. Following Dym & Maron [18], we split the model into two parts: Embedding functions $\mathcal{F}_{\text{feat}}$ that lifts the input into an equivariant representation, and pooling functions $\mathcal{F}_{\text{pool}}$ that aggregate the results of multiple embedding functions on each point and computes the model output. The overall model is then defined as the set of functions

$$\mathcal{F}^{\text{TFN}}_{K(D),D} = \{f \mid f(\boldsymbol{X}, \boldsymbol{H}_{\text{in}}) = \sum_{k=1}^{K} f^{(k)*}_{\text{pool}}(f^{(k)}_{\text{feat}}(\boldsymbol{X}, \boldsymbol{H}_{\text{in}})), f^{(k)}_{\text{pool}} \in \mathcal{F}^{\text{TFN}}_{\text{pool}}(D), f^{(k)}_{\text{feat}} \in \mathcal{F}^{\text{TFN}}_{\text{feat}}(D)\}, \tag{1}$$

where $D \in \mathbb{N}$ denotes the function's maximum polynomial degree, $K(D) \in \mathbb{N}$ is chosen such that Theorem 1 is fulfilled (Dym & Maron [18] only prove the existence of this function), and f^* denotes elementwise application of f on all points. We then define the set $\mathcal{F}^{\text{TFN}}_{\text{pool}}$ as all rotationally equivariant linear functions on the $\mathrm{SO}(3)$ group, i.e. all $\mathrm{SO}(3)$ convolutions [38]. Note that these are more expressive than the self-interaction layers used originally [56]. The embedding functions $\mathcal{F}^{\text{TFN}}_{\text{feat}}(D) = \{\pi_2 \circ f^{(2D)} \circ \cdots \circ f^{(1)} \mid f^{(i)} \in \mathcal{F}^{\text{TFN}}_{\text{prod}}\}$ consist of an auxiliary function $\pi_2(\boldsymbol{X}, \boldsymbol{H}) = \boldsymbol{H}$ and a series of tensor product functions (called convolution by Dym & Maron [18]) $\mathcal{F}^{\text{TFN}}_{\text{prod}} = \{f \mid f(\boldsymbol{X}, \boldsymbol{H}) = (\boldsymbol{X}, \tilde{\boldsymbol{H}}^{\text{TFN}}(\boldsymbol{X}, \boldsymbol{H}))\}$. The intermediate representations are $\boldsymbol{H} \in W^n_{\text{feat}}$, where W_{feat} is a representation of $\mathrm{SO}(3)$ indexed by the degree l and the order m. For $\boldsymbol{H}_{\text{in}}$ we have $l{=}m{=}0$. The main update is defined by

$$\tilde{\boldsymbol{H}}^{\text{TFN}(l_o)}_{am_o}(\boldsymbol{X}, \boldsymbol{H}) = \theta \boldsymbol{H}^{(l_o)}_{am_o} + \sum_{l_f, m_f} \sum_{l_i, m_i} C^{(l_o, m_o)}_{(l_f, m_f), (l_i, m_i)} \sum_{b \in \mathcal{N}_a} F^{(l_f)}_{\text{TFN}, m_f}(\boldsymbol{x}_b - \boldsymbol{x}_a) \boldsymbol{H}^{(l_i)}_{bm_i}, \tag{2}$$

where θ is a (learned) scalar and $\mathcal{N}_a$ are the neighbors of point a. The Clebsch-Gordan coefficients $C^{(l_o, m_o)}_{(l_f, m_f), (l_i, m_i)}$ arise from decomposing the tensor product of two input $\mathrm{SO}(3)$ representations (the filter and input representations) into a sum of output representations. Their exact values are not relevant for this discussion. We index the output with degree l_o and order m_o, the learned filter with l_f and m_f, and the input with l_i and m_i. $F^{(l)}_{\text{TFN}, m}(\boldsymbol{x}) = R^{(l)}(x) Y_{lm}(\hat{\boldsymbol{x}})$ is a rotationally equivariant

filter, with a (learned) radial part R, which is any polynomial of degree $\leq D$, and the real spherical harmonics Y_{lm} with degree l and order m. The spherical harmonics are the basis for the Fourier transformation of functions on the sphere, analogously to sine waves for functions on $\mathbb{R}$. We can prove universality for TFNs by using the universality of polynomial regression and showing that TFNs can fit any polynomial (see Dym & Maron [18] for details), resulting in:

Theorem 1 (Dym & Maron [18])**.** *Consider the set of functions $\mathcal{G}$ mapping $\mathbb{R}^{3\times n+h\times n} \to W_{\mathrm{T}}^n$ that are equivariant to rotations and permutations and invariant to translations. For all $n \in \mathbb{N}$,*

1. *For $D \in \mathbb{N}_0$, every polynomial $p \in \mathcal{G}$ of degree D is in $\mathcal{F}^{\text{TFN}}_{K(D),D}$.*
2. *Every continuous function $f \in \mathcal{G}$ can be approximated uniformly on compact sets by functions in $\bigcup_{D\in\mathbb{N}_0} \mathcal{F}^{\text{TFN}}_{K(D),D}$.*

Spherical networks. Instead of intermediate $\mathrm{SO}(3)$ representations we now switch to spherical representations, which are functions on the sphere $\boldsymbol{H} : S^2 \to \mathbb{R}$. We define the set of functions $\mathcal{F}^{\text{sphere}}_{K(D),D}$ analogously to $\mathcal{F}^{\text{TFN}}_{K(D),D}$. However, for $\mathcal{F}^{\text{sphere}}_{\text{feat}}(D)$ we use

$$\tilde{\boldsymbol{H}}_a^{\text{sphere}}(\boldsymbol{X}, \boldsymbol{H})(\hat{\boldsymbol{r}}) = \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}) + \sum_{b\in\mathcal{N}_a} F_{\text{sphere}}(\boldsymbol{x}_b - \boldsymbol{x}_a, \hat{\boldsymbol{r}}) \boldsymbol{H}_b(\hat{\boldsymbol{r}}), \tag{3}$$

with the filter function $F_{\text{sphere}}(\boldsymbol{x}, \hat{\boldsymbol{r}}) = \sum_{l,m} R^{(l)}(x) \Re[Y_m^{(l)*}(\hat{\boldsymbol{x}}) Y_m^{(l)}(\hat{\boldsymbol{r}})]$, using the real part $\Re$ of the complex spherical harmonics $Y_m^{(l)}$. The set of pooling functions for invariant predictions is

$$\mathcal{F}^{\text{sphere}}_{\text{pool}} = \{f \mid f(\boldsymbol{H}) = \theta_{\text{pool}} \int_{S^2} \boldsymbol{H}(\hat{\boldsymbol{r}})\, \mathrm{d}\hat{\boldsymbol{r}}\}, \tag{4}$$

with the learnable parameter θ_{pool}. We obtain the universality theorem by showing the equivalence between this model and TFN for rotationally invariant functions. The proof is based on the connection between spherical harmonics and the Clebsch-Gordan coefficients [51, 3.7.72] (see App. A).

Theorem 2. *Consider the set of functions $\mathcal{G}'$ mapping $\mathbb{R}^{3\times n+h\times n} \to W_{\mathrm{T}}^n$ that are equivariant to permutations and invariant to translations and rotations. For all $n \in \mathbb{N}$,*

1. *For $D \in \mathbb{N}_0$, every polynomial $p \in \mathcal{G}'$ of degree D is in $\mathcal{F}^{\text{sphere}}_{K(D),D}$.*
2. *Every continuous function $f \in \mathcal{G}'$ can be approximated uniformly on compact sets by functions in $\bigcup_{D\in\mathbb{N}_0} \mathcal{F}^{\text{sphere}}_{K(D),D}$.*

Next, we extend Theorem 2 to rotationally *equivariant* functions. We do this by generalizing a recent result by Villar et al. [57] to obtain (see App. B):

Theorem 3. *Let $h\colon \mathbb{R}^{d\times n+h\times n} \to \mathbb{R}^{d\times n}$ be any function that is equivariant to permutations and rotations and invariant to translations. For all $a \in [1, n]$, let the set of relative vectors $\{\boldsymbol{x}_{ca} \mid c \in [1, n]\}$ not span a $(d-1)$-dimensional space. Then there are $n-1$ functions $f^{(c)}\colon \mathbb{R}^{d\times n+h\times n} \to \mathbb{R}^n$ such that*

$$\boldsymbol{h}_a(\boldsymbol{X}, \boldsymbol{H}) = \sum_{\substack{c=1\\ c\neq a}}^{n} f_a^{(c)}(\boldsymbol{X}, \boldsymbol{H}) \boldsymbol{x}_{ca}, \tag{5}$$

where $f^{(c)}$ is equivariant to permutations, but invariant to rotations and translations.

This theorem lets us extend a rotationally invariant model to an equivariant one, while providing universality guarantees. Together, Theorem 2 and Theorem 3 (with $d = 3$) thus show that we can approximate any rotationally equivariant function using only representations on the S^2 sphere. We thus do not need $\mathrm{SO}(3)$ representations, spin-weighted spherical harmonics [22], triplet embeddings, or complex-valued functions. This result puts theory back in line with practice, where the best results are currently achieved without relying on these more expensive representations [54].

4 From spherical representations to directional message passing

Directional representations. To use spherical representations in a model we first need to find a tractable description. Instead of using spherical harmonics, we propose to sample the representations

in specific directions $\hat{\boldsymbol{r}}_i$. If we look at recent models, we see that they implicitly use the directions to each atom's neighbors for this purpose, i.e. they embed the edges in the molecule's graph. These directions define an *equivariant* mesh that circumvents the aliasing effects that would arise from fixed grids [36]. Schütt et al. [54] flexibly define the directional mesh in each layer by aggregating directions, while Klicpera et al. [35] and others use a fixed mesh for each atom. We can refine this mesh of directions e.g. by using more neighbors or by interpolating between directions. The approximation error of this directional mesh is related to the spherical harmonic expansion via the mesh norm and the separating distance between directions [31, 32]. Note that depending on the discretization scheme the resulting mesh might not provide a universal approximation guarantee.

Eq. (3) only defines the relationship for a fixed direction, while models commonly use different directional meshes for the input and output. To incorporate this we add a convolution with a learned filter F_2, which can only improve the model's expressiveness. Since the input and output are spherical functions, the used filter F_2 has to be *zonal*, i.e. it can only depend on one angle. This can be expressed as [17]

$$\begin{aligned}\tilde{\boldsymbol{H}}_a^{\text{dir}}(\boldsymbol{X},\boldsymbol{H})(\hat{\boldsymbol{r}}_o) &= \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}_o) + \int_{\text{SO}(3)} \sum_{b\in\mathcal{N}_a} F_{\text{sphere}}(\boldsymbol{x}_{ba}, \boldsymbol{R}\hat{\boldsymbol{n}}) \sum_{i\in\mathcal{R}_b} \boldsymbol{H}_{bi}\delta(\boldsymbol{R}\hat{\boldsymbol{n}} - \hat{\boldsymbol{r}}_i) F_2(\boldsymbol{R}^{-1}\hat{\boldsymbol{r}}_o)\,\mathrm{d}\boldsymbol{R} \\ &= \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}_o) + \sum_{b\in\mathcal{N}_a}\sum_{i\in\mathcal{R}_b} F_{\text{sphere}}(\boldsymbol{x}_{ba}, \hat{\boldsymbol{r}}_i)\boldsymbol{H}_{bi} F_2(\measuredangle \hat{\boldsymbol{r}}_o\hat{\boldsymbol{r}}_i),\end{aligned} \tag{6}$$

where $\mathcal{R}_b$ denotes the directional mesh of atom b with mesh directions denoted by $\hat{\boldsymbol{r}}_i$, and $\hat{\boldsymbol{r}}_o$ specifies the output direction. The integral vanishes due to the Dirac delta δ.

General filters. To see the relationship to GNNs we furthermore need to generalize the filter $F_{\text{sphere}}(\boldsymbol{x}_{ba}, \hat{\boldsymbol{r}}_i)$. This filter only depends on the angle $\measuredangle \hat{\boldsymbol{r}}_i\hat{\boldsymbol{x}}_{ba}$ since it is rotationally invariant:

Lemma 1. $F_{\text{sphere}}(\boldsymbol{R}\boldsymbol{x}, \boldsymbol{R}\hat{\boldsymbol{r}}) = F_{\text{sphere}}(\boldsymbol{x}, \hat{\boldsymbol{r}})$ *for any rotation matrix* $\boldsymbol{R}$.

We can therefore substitute F_{sphere} with a general learnable filter F_1 that is parametrized by this relative angle. Since F_{sphere} arises as a special case we do not lose expressivity. We thus obtain

$$\tilde{\boldsymbol{H}}_a^{\text{gem}}(\boldsymbol{X},\boldsymbol{H})(\hat{\boldsymbol{r}}_o) = \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}_o) + \sum_{b\in\mathcal{N}_a}\sum_{i\in\mathcal{R}_b} F_1(x_{ba}, \measuredangle \hat{\boldsymbol{r}}_i\hat{\boldsymbol{x}}_{ba}) F_2(\measuredangle \hat{\boldsymbol{r}}_o\hat{\boldsymbol{r}}_i)\boldsymbol{H}_{bi}. \tag{7}$$

We have now arrived at a message passing scheme that has universal approximation guarantees and is only based on relative directional information. To see the connection to GNNs we interpret these discretized spherical representations as edge embeddings pointing towards $\hat{\boldsymbol{r}}_o$ and $\hat{\boldsymbol{r}}_i$. Eq. (7) then corresponds to two-hop message passing between the edge embeddings of $\hat{\boldsymbol{r}}_o$ and $\hat{\boldsymbol{r}}_i$ via the edge $\hat{\boldsymbol{x}}_{ba}$. Interestingly, the central learnable part of Eq. (7) is the product of the filters $F_1(x_{ba}, \measuredangle \hat{\boldsymbol{r}}_i\hat{\boldsymbol{x}}_{ba})$ and $F_2(\measuredangle \hat{\boldsymbol{r}}_o\hat{\boldsymbol{r}}_i)$ with the input representation, which is strikingly similar to the Hadamard product used in modern GNNs [34, 53] – except that these only use one-hop message passing.

5 Geometric message passing

Geometric representation. We now develop a specific two-hop message passing scheme based on Eq. (7). We use embeddings based on interatomic directions, and embed all atom pairs with distance $x_{ca} \leq c_{\text{emb}}$. $\hat{\boldsymbol{r}}_o$ and $\hat{\boldsymbol{r}}_i$ are thus instantiated as the interatomic directions $\hat{\boldsymbol{x}}_{ca}$ and $\hat{\boldsymbol{x}}_{db}$. We denote directional embeddings as $\boldsymbol{m}_{ca} = \boldsymbol{H}_a(\hat{\boldsymbol{x}}_{ca})$. Message passing is thus based on quadruplets of atoms – two atoms are interacting (a and b) and two atoms define the directions (c and d). We denote the angle between directions by $\varphi_{abd} = \measuredangle \hat{\boldsymbol{x}}_{ab}\hat{\boldsymbol{x}}_{db}$. To improve empirical performance we additionally use the dihedral angle $\theta_{cabd} = \measuredangle \hat{\boldsymbol{x}}_{ca}\hat{\boldsymbol{x}}_{db} \perp \hat{\boldsymbol{x}}_{ba}$ and substitute $\measuredangle \hat{\boldsymbol{r}}_o\hat{\boldsymbol{r}}_i = \measuredangle \hat{\boldsymbol{x}}_{ca}\hat{\boldsymbol{x}}_{db}$ with φ_{cab}. Fig. 1 illustrates the three angles φ_{cab}, φ_{abd}, and θ_{cabd} we use for updating the embedding $\boldsymbol{m}_{ca}$ based on $\boldsymbol{m}_{db}$. To ensure that all angles are well-defined we exclude overlapping atom quadruplets, i.e. $a \neq b \neq c \neq d$. We represent the relative directional information using spherical Fourier-Bessel

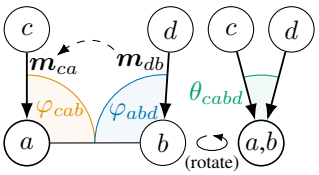

Figure 1: Angles used in geometric message passing. The dihedral angle θ_{cabd} becomes visible when rotating the molecule so that atoms a and b lie on top of each other (right).

bases with polynomial radial envelopes to ensure smoothly differentiable predictions, as proposed by Klicpera et al. [35]. We split the basis into three parts to incorporate all available geometric information. Before the envelope, these are:

$$\tilde{\boldsymbol{e}}_{\text{RBF},n}(x_{db}) = \sqrt{\frac{2}{c_{\text{emb}}}} \frac{\sin(\frac{n\pi}{c_{\text{emb}}} x_{db})}{x_{db}}, \tag{8}$$

$$\tilde{\boldsymbol{e}}_{\text{CBF},ln}(x_{ba}, \varphi_{abd}) = \sqrt{\frac{2}{c_{\text{int}}^3 j_{l+1}^2(z_{ln})}} j_l(\frac{z_{ln}}{c_{\text{int}}} x_{ba}) Y_{l0}(\varphi_{abd}), \tag{9}$$

$$\tilde{\boldsymbol{e}}_{\text{SBF},lmn}(x_{ca}, \varphi_{cab}, \theta_{cabd}) = \sqrt{\frac{2}{c_{\text{emb}}^3 j_{l+1}^2(z_{ln})}} j_l(\frac{z_{ln}}{c_{\text{emb}}} x_{ca}) Y_{lm}(\varphi_{cab}, \theta_{cabd}), \tag{10}$$

with the interaction cutoff c_{int}, the spherical Bessel functions j_l, and the n-th root of the l-order Bessel function z_{ln}. Note that Klicpera et al. [35] only used the first two parts $\boldsymbol{e}_{\text{RBF}}$ and $\boldsymbol{e}_{\text{CBF}}$. These representations are then transformed using two linear layers to obtain the filter F. In order to maintain a smoothly differentiable cutoff we cannot use a bias in this transformation. Altogether, the core geometric message passing scheme is

$$\begin{aligned} \tilde{\boldsymbol{m}}_{ca} = \sum_{\substack{b \in \mathcal{N}_a^{\text{int}} \setminus \{c\}, \\ d \in \mathcal{N}_b^{\text{emb}} \setminus \{a,c\}}} & \Big((\boldsymbol{W}_{\text{SBF1}} \boldsymbol{e}_{\text{SBF}}(x_{ca}, \varphi_{cab}, \theta_{cabd}))^T \mathbf{W} ((\boldsymbol{W}_{\text{CBF2}} \boldsymbol{W}_{\text{CBF1}} \boldsymbol{e}_{\text{CBF}}(x_{ba}, \varphi_{abd})) \\ & \odot (\boldsymbol{W}_{\text{RBF2}} \boldsymbol{W}_{\text{RBF1}} \boldsymbol{e}_{\text{RBF}}(x_{db})) \odot \boldsymbol{m}_{db}) \Big), \end{aligned} \tag{11}$$

where $\boldsymbol{W}$ denotes a weight matrix, $\mathbf{W}$ denotes a weight tensor. The first weight matrix of each representation part has a small output dimension. This causes a bottleneck that improves generalization.

Symmetric message passing. Whenever we have a directional embedding $\boldsymbol{m}_{ca}$, we also have the opposing embedding $\boldsymbol{m}_{ac}$, since both are based on the same cutoff c_{emb}. Whether we associate the embedding $\boldsymbol{m}_{ca}$ or $\boldsymbol{m}_{ac}$ with atom a is arbitrary. A more principled approach is to *jointly* interpret both embeddings as a representation of the atom pair a and c. In this view, an update to $\boldsymbol{m}_{ca}$ should also influence $\boldsymbol{m}_{ac}$. This would normally require executing the above message passing scheme twice, once for updating $\boldsymbol{m}_{ca}$ based on $\boldsymbol{m}_{db}$, and once for updating $\boldsymbol{m}_{ac}$ based on $\boldsymbol{m}_{db}$. We propose to circumvent this double execution by calculating the update (Eq. (11)) only once and then using it for both $\boldsymbol{m}_{ca}$ and $\boldsymbol{m}_{ac}$. To preserve the distinction between the two directions and ensure that $\boldsymbol{m}_{ca} \neq \boldsymbol{m}_{ac}$, we transform the two updates using two separate learnable weight matrices. One single message passing update thus carries information for both embeddings, which is then dissected by the two weight matrices. In practice, this only requires a simple re-indexing operation that maps the edge ca to ac.

Efficient bilinear layer. The whole message passing scheme, i.e. basis transformation, neighbor aggregation, and bilinear layer, only use linear functions. We can therefore freely optimize the order of summation without changing the result, as proposed by Wu et al. [59] (see App. D for details). Doing so can provide a faster and more memory-efficient model, reducing memory usage by 50 % even for Hadamard products. Moreover, since everything is based on efficient matrix products, this allows us to use the bilinear layer at practically no additional cost compared to a Hadamard product. Note that this requires using padded matrices instead of the usual gather-scatter operations to prevent excessively large intermediate results.

6 GemNet: Geometric message passing neural network

GemNet. The geometric message passing neural network (GemNet) is a significantly refined architecture based on DimeNet^{++} [34, Hippocratic license 2.1]. GemNet predicts the molecular energy E and forces $\boldsymbol{F} \in \mathbb{R}^{3\times n}$ based on the atomic positions $\boldsymbol{X} \in \mathbb{R}^{3\times n}$ and the atomic numbers $\boldsymbol{z} \in \mathbb{N}^n$. The architecture is illustrated in Fig. 2. A comprehensive version with low-level layers and hyperparameters is described in App. F. GemNet was developed on the COLL dataset, but generalizes to other datasets such as MD17 without architectural changes. Every change we propose either improves model performance or reduces model complexity. For example, GemNet uses no biases since we found them to be irrelevant or even detrimental to accuracy. We show the impact of the most relevant changes via ablation studies in Sec. 7.

Interactions. GemNet incorporates three forms of interactions. The first is geometric message passing, as described in Sec. 5. The second is a one-hop form of geometric message passing. This

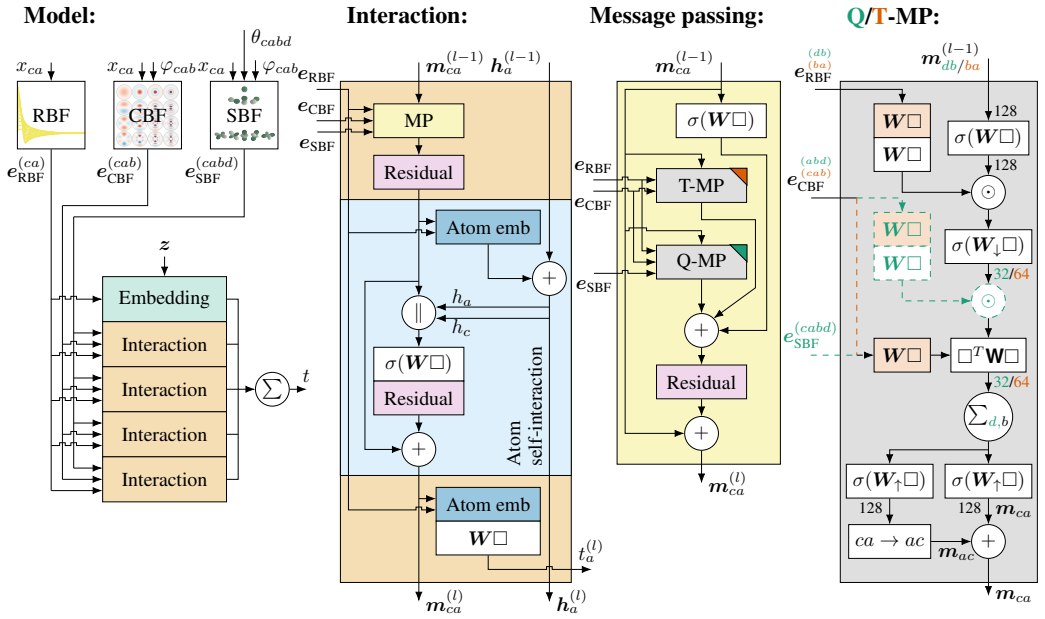

Figure 2: The GemNet architecture (comprehensive version in App. F). $\square$ denotes the layer's input, $\|$ concatenation, and σ a non-linearity. Directional embeddings $\boldsymbol{m}_{ca}$ are updated using three forms of interaction: Two-hop geometric message passing (Q-MP), one-hop geometric message passing (T-MP), and atom self-interactions. Differences between Q-MP and T-MP are denoted by colors and dashed lines.

interaction uses a single cutoff $c = c_{\text{emb}}$ and passes messages only between directional embeddings pointing towards the same atom, similarly to DimeNet [35]. This provides both angle-based pair interactions and atom self-interactions, thanks to the symmetric message passing scheme described in Sec. 5. The third interaction is a pure atom self-interaction based on atom embeddings. We first aggregate the directional embeddings of one atom to obtain an atom embedding. We then use this atom embedding to update all directional embeddings. We found all three interaction forms to be beneficial, and show this in our ablation studies.

Stabilizing activation variance. The variance of activations in a model is usually stabilized using normalization methods, which has various positive effects on training [16, 41, 52]. However, they also have multiple undesirable side effects, especially in the context of molecular regression. Batch normalization introduces correlations between separate molecules and atoms. Layer normalization forces all activation scales to be constant, while atomic interactions actually cover a large range of scales – directly bonding atoms have a substantially stronger interaction than atoms at a long range. To circumvent these issues, we stabilize GemNet's variance by introducing constant scaling factors, as suggested by Brock et al. [6]. We found that the activation variance is primarily impacted by four components: Skip connections, non-linearities, message aggregation, and Hadamard/bilinear layers. The two summands in a skip connection $y = x + f(x)$ have no covariance at initialization due to random weight matrices. We can thus remove its impact by scaling the output by $1/\sqrt{2}$. We remove the non-linearity's impact by scaling its output with a gain of $\gamma = 1/0.6$ for SiLU, similarly to [33]. Note that we do not center SiLU's output but instead choose a slightly lower γ to account for mean shift. Additionally, we standardize the weight matrices to have exactly zero mean and $1/\text{fan-in}$ variance. The sum aggregation and Hadamard/bilinear layers have a more complex impact on the variance, which we cannot determine a priori (see App. E for details). We therefore estimate the variance after these layers based on random batches of data. We then rescale their output accordingly to obtain roughly the variance of the layer input at initialization. These simple empirical scaling factors are sufficient to keep the activation variance roughly constant (see Fig. 3). We found that other measures such as adaptive gradient clipping [7], scaled weight standardization [6], or weighting the residual block with zero at initialization [16] are not beneficial for model accuracy.

Table 1: MAE on COLL, in meV/Å and meV. GemNet is 34 % more accurate for forces. The higher energy error is due to its lower loss weight.

	Forces	Energy
SchNet	172	198
DimeNet^{++}	40	**47**
GemNet-Q	**26.4**	53
GemNet-T	31.6	60
GemNet-dQ	38.1	60
GemNet-dT	43.1	55

Table 2: Force MAE for MD17@CCSD in meV/Å. GemNet outperforms previous methods by 44 % on average.

	sGDML	NequIP	GemNet-Q	GemNet-T
Aspirin	33.0	14.7	10.4	**10.3**
Benzene	1.7	0.8	**0.7**	**0.7**
Ethanol	15.2	9.4	**3.1**	**3.1**
Malonaldehyde	16.0	16.0	6.0	**5.9**
Toluene	9.1	4.4	**2.5**	2.7

GemNet-Q and GemNet-T. Geometric message passing is comparatively expensive since it is based on quadruplets of atoms. Its runtime thus scales with $\mathcal{O}(nk_{\text{int}}k_{\text{emb}}^2)$, where k_{int} is the number of interacting neighbors, and k_{emb} is the number of embedded directions. For this reason we investigate two message passing models in our experiments – one with two-hop geometric message passing (GemNet-Q) and one using only the two cheaper forms of interaction (GemNet-T). Their complexities are $\mathcal{O}(nk_{\text{int}}k_{\text{emb}}^2)$ and $\mathcal{O}(nk_{\text{emb}}^2)$, respectively. Note that GemNet-T is thus a direct ablation of the two-hop message passing scheme implied by our theoretical results.

Direct force predictions. GemNet predicts forces by calculating $\boldsymbol{F}_a = -\partial E / \partial \boldsymbol{x}_a$ via backpropagation. This form of calculation guarantees a conservative force field, which is important for the stability of simulations. However, by using Eq. (5) we can also directly predict forces and other vector quantities. This essentially means predicting a magnitude for each directional embedding and then summing up over the vectors defined by this magnitude and the embedding's associated direction, similarly to Park et al. [47]. We denote this variant as *GemNet-dQ* and *GemNet-dT*. Interestingly, GemNet is thus able to generate rotationally *equivariant* predictions despite only using *invariant* representations. Direct predictions substantially accelerate the model, especially for training. For most datasets, the resulting accuracy is on par with most previous models, but significantly worse than GemNet's accuracy via backpropagation. However, this is not true for OC20, where we found GemNet-dT to converge faster and perform on par with GemNet-T.

Limitations. GemNet is focused on one specific, important task: Predictions for molecular simulations. We do not make any statements regarding its performance beyond this scope. The GemNet architecture might seem more complex than some previous models, due to its larger variety of interactions and blocks. However, its number of parameters and training or inference time is actually on par with previous models. Two-hop message passing introduces significant computational overhead. We mitigate this effect with a down-projection layer and additionally introduce the ablated GemNet-T model. This model performs surprisingly well on MD17, but not on COLL. This suggests that one-hop message passing is expressive enough for some practical use cases, but two-hop message passing gives an advantage for the more challenging task of fitting multiple molecules at once.

Societal impacts. Accelerating molecular simulations can have positive effects in a wide range of applications in physics and chemistry. At the same time, however, this can be used for malicious purposes such as developing chemical agents or weapons. To the best of our knowledge, this work does not promote these use cases more than regular chemistry research does. To somewhat mitigate negative effects we will publish our source code under the Hippocratic license [19].

7 Experiments

Experimental setup. We evaluate our model on four molecular dynamics datasets. COLL [34, CC-BY 4.0] consists of configurations taken from molecular collisions of different small organic molecules. MD17 [9] consists of configurations of multiple separate, thermalized molecules, considering only one molecule at a time. MD17@CCSD [10] uses the same setup, but calculates the forces using the more accurate and expensive CCSD or CCSD(T) method. The open catalyst (OC20) dataset [8, CC-BY 4.0] consists of energy relaxation trajectories of solid catalysts with adsorbate molecules. This dataset is split into three tasks: (1) Structure to energy and forces (S2EF), which is the same task as used by the COLL and MD17 datasets, (2) initial structure to relaxed structure (IS2RS), where

Table 3: Force MAE for MD17 in meV/Å. GemNet outperforms all previous methods by a wide margin, on average by 41 %.

	Kernel methods		GNNs				**GemNet**	
	sGDML	FCHL19	DimeNet	SphereNet	NequIP	PaiNN	GemNet-Q	GemNet-T
Aspirin	29.5	20.7	21.6	18.6	15.1	14.7	**9.4**	9.5
Benzene[9]	-	-	8.1	7.7	8.1	-	**6.3**	**6.3**
Benzene[10]	2.6	-	-	-	2.3	-	1.5	**1.4**
Ethanol	14.3	5.9	10.0	9.0	9.0	9.7	3.8	**3.7**
Malonaldehyde	17.8	10.6	16.6	14.7	14.6	14.9	6.9	**6.7**
Naphthalene	4.8	6.5	9.3	7.7	4.2	3.3	**2.2**	2.4
Salicylic acid	12.1	9.6	16.2	15.6	10.3	8.5	**5.4**	5.5
Toluene	6.1	8.8	9.4	6.7	4.4	4.1	**2.6**	**2.6**
Uracil	10.4	4.6	13.1	11.6	7.5	6.0	4.5	**4.2**

Table 4: Results for the three tasks of the open catalyst dataset (OC20), averaged across its four test sets. GemNet outperforms all previous methods in all measures, on average by 20 %.
*DimeNet^{++}-large uses separate models for energy and force prediction for IS2RE.

	S2EF			IS2RS		IS2RE
	Energy MAE meV ↓	Force MAE meV/Å ↓	Force cos ↑	AFbT % ↑	ADwT % ↑	Energy MAE meV ↓
ForceNet-large	-	31.2	0.520	12.7	49.6	-
DimeNet^{++}-large*	-	31.3	0.544	21.8	51.7	559.1
SpinConv	336.3	29.7	0.539	16.7	53.6	434.3
GemNet-dT	**292.4**	**24.2**	**0.616**	**27.6**	**58.7**	**399.7**

an energy optimization is carried out based on the model's predictions and we measure how close the final structure is to the true relaxed structure (average distance within threshold, ADwT) and whether the final forces are close to zero (average forces below threshold, AFbT), and (3) initial structure to relaxed energy (IS2RE), where we predict the energy of the relaxed structure, based on an energy optimization starting at the initial structure. All presented OC20 models are trained on the S2EF data. Following the setup of Batzner et al. [4], we use 1000 training and validation configurations for MD17, and 950 training and 50 validation configurations for MD17@CCSD. We focus on force predictions and use a high force loss weight since they determine the accuracy of molecular simulations. We measure the mean absolute error (MAE), averaged over all samples, atoms, and components. We compare with the results reported by several state-of-the-art models: sGDML [10], FCHL19 [12], SchNet [53], DimeNet [35], DimeNet^{++} [34], SphereNet [39], NequIP [4], PaiNN [54], ForceNet [30], and SpinConv [55]. For further details see App. G.

Results. Tables 1 to 4 show that GemNet-T and GemNet-Q consistently perform best on all molecular dynamics datasets investigated – and by a large margin. This is true both in comparison to previous GNNs and for kernel methods – despite the latter typically being more sample efficient. The improvements are largest for chain-like molecules, such as ethanol and malonaldehyde. These molecules are the most challenging since they exhibit a wide range of movement. GemNet even performs better than some previous models that were trained with 50x more training samples. For example, it performs better than SchNet with 50 000 training samples on six out of eight MD17 molecules (see Table 9). Interestingly, the two-hop message passing scheme implied by our theoretical results (GemNet-Q) yields significant improvements on COLL, but performs approximately on par with the ablated GemNet-T on MD17. To investigate this disagreement we trained GemNet on a combined dataset of all MD17 molecules. Table 13 shows that GemNet-Q again performs better than GemNet-T in this setting. These results suggest that regular MD17 is too simple to show the benefits of two-hop message passing. It seems to be particularly important in more difficult settings that cover a large variety of configurations and molecules.

Computational aspects. GemNet-Q is roughly two times slower than GemNet-T (see Table 14). Thanks to the efficient aggregation, GemNet with bilinear layers is as fast as with regular Hadamard products. Efficient aggregation also reduces the memory usage for regular Hadamard products by around 50 % (from 4.1GB to 2.2GB for a batch of 32 Toluene molecules). Note that GemNet has not

been optimized for runtime and can likely be accelerated substantially. GemNet-Q uses 2.2M and GemNet-T 1.9M parameters, which is comparable to previous models such as DimeNet^{++}, which uses 1.9M parameters. See App. I for further details.

Direct force prediction. Directly predicting the forces accelerates training by four times on average and inference by 1.6 times on average in our experiments (see Table 14), while reducing memory consumption by roughly a factor of two. While using direct predictions instead of backpropagation increases the MAE by 44 % on COLL and by 48 % on MD17 (see Tables 1 and 7), they actually perform better on the S2EF task on OC20. This is likely due to OC20 being orders of magnitude larger than COLL and MD17. Whether to use direct predictions thus depends on the dataset and the application's computational requirements.

Ablation studies. We investigate the proposed architectural improvements on COLL in Table 5. The proposed symmetric message passing scheme yields significant accuracy improvements, as does using a bilinear layers instead of a Hadamard product. We also see that removing any of the three interaction forms described in Sec. 6 increases the error, showing that this combination is indeed beneficial. The proposed scaling factors also yield decent improvements, while regular layer normalization actually increases the error. Two-hop message passing yields the largest single improvement. Table 10 shows that our architectural improvements yield similar benefits for DimeNet^{++}. Overall, the error improvements are quite evenly distributed. This suggests that GemNet's improved performance is not due to one single change, but rather due to the full range of improvements proposed in this work.

Table 5: Ablation studies on COLL. Force MAE in meV/Å after 500 000 training steps. All proposed components yield significant improvements.

Model	Forces
without symmetric message passing	28.5
Hadamard product instead of bilinear layer	29.3
without atom embedding updates	28.3
without one-hop message passing	31.3
without two-hop message passing	32.4
without scaling factors	29.1
use layer norm instead (without centering)	33.3
with bias	27.2
GemNet-Q	27.0

8 Conclusion

In this work we proved the universality for GNNs using directional embeddings. We proposed geometric message passing based on these insights, and improved this scheme with symmetric message passing and efficient bilinear layers. We incorporated these improvements in the GemNet architecture, which substantially improves the error on various molecular dynamics datasets. We showed that all of the proposed enhancements yield significant performance improvements. Most of our proposed improvements are of independent interest for other molecular GNNs.

Acknowledgments and Disclosure of Funding

We would like to thank Soledad Villar for help with proving Lemma B, as well as Nicholas Gao and Aleksandar Bojchevski for their invaluable feedback and comments.

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) through the Emmy Noether grant GU 1409/2-1 and the TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

References

[1] Brandon M. Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant Molecular Neural Networks. In *NeurIPS*, 2019.

[2] Waiss Azizian and Marc Lelarge. Expressive Power of Invariant and Equivariant Graph Neural Networks. In *ICLR*, 2020.

[3] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Physical Review Letters*, 104(13):136403, 2010.

[4] Simon Batzner, Tess E. Smidt, Lixin Sun, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, and Boris Kozinsky. SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *arXiv*, 2101.03164, 2021.

[5] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz Group Equivariant Neural Network for Particle Physics. In *ICML*, 2020.

[6] Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *ICLR*, 2021.

[7] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-Performance Large-Scale Image Recognition Without Normalization. *arXiv*, 2102.06171, 2021.

[8] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open Catalyst 2020 (OC20) Dataset and Community Challenges. *ACS Catalysis*, 11(10):6059–6072, 2021.

[9] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[10] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature Communications*, 9(1):1–10, 2018.

[11] 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.

[12] Anders S. Christensen, Lars A. Bratholm, Felix A. Faber, and O. Anatole von Lilienfeld. FCHL revisited: Faster and more accurate quantum machine learning. *The Journal of Chemical Physics*, 152(4):044107, 2020.

[13] Taco Cohen and Max Welling. Group Equivariant Convolutional Networks. In *ICML*, 2016.

[14] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *ICLR*, 2018.

[15] Taco S Cohen, Mario Geiger, and Maurice Weiler. A General Theory of Equivariant CNNs on Homogeneous Spaces. In *NeurIPS*, 2019.

[16] Soham De and Sam Smith. Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks. In *NeurIPS*, 2020.

[17] J. R. Driscoll and D. M. Healy. Computing Fourier Transforms and Convolutions on the 2-Sphere. *Advances in Applied Mathematics*, 15(2):202–250, 1994.

[18] Nadav Dym and Haggai Maron. On the Universality of Rotation Equivariant Point Cloud Networks. In *ICLR*, 2021.

[19] Coraline Ada Ehmke. The Hippocratic License 2.1: An Ethical License for Open Source., 2020. URL `https://firstdonoharm.dev`.

[20] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[21] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) Equivariant Representations with Spherical CNNs. In *ECCV*, 2018.

[22] Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. Spin-Weighted Spherical CNNs. In *NeurIPS*, 2020.

[23] Felix A. Faber, Anders S. Christensen, Bing Huang, and O. Anatole von Lilienfeld. Alchemical and structural distribution based representation for universal quantum machine learning. *The Journal of Chemical Physics*, 148(24):241717, 2018.

[24] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data. In *ICML*, 2020.

[25] Daniel Flam-Shepherd, Tony C. Wu, Pascal Friederich, and Alan Aspuru-Guzik. Neural Message Passing on High Order Paths. *Machine Learning: Science and Technology*, 2021.

[26] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *NeurIPS*, 2020.

[27] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and Representational Limits of Graph Neural Networks. In *ICML*, 2020.

[28] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, 2017.

[29] Thomas A. Halgren. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *Journal of Computational Chemistry*, 17(5-6):490–519, 1996.

[30] Weihua Hu, Muhammed Shuaibi, Abhishek Das, Siddharth Goyal, Anuroop Sriram, Jure Leskovec, Devi Parikh, and C. Lawrence Zitnick. ForceNet: A Graph Neural Network for Large-Scale Quantum Calculations. *arXiv*, 2103.01436, 2021.

[31] Kurt Jetter, Joachim Stöckler, and Joseph Ward. Error estimates for scattered data interpolation on spheres. *Mathematics of Computation*, 68(226):733–747, 1999.

[32] Jens Keiner, Stefan Kunis, and Daniel Potts. Efficient Reconstruction of Functions on the Sphere from Scattered Data. *Journal of Fourier Analysis and Applications*, 13(4):435–458, 2007.

[33] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NeurIPS*, 2017.

[34] Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules. In *Machine Learning for Molecules Workshop, NeurIPS*, 2020.

[35] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *ICLR*, 2020.

[36] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network. In *NeurIPS*, 2018.

[37] Risi Kondor, Truong Son Hy, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. Covariant Compositional Networks For Learning Graphs. In *International Workshop on Mining and Learning with Graphs (MLG), KDD*, 2019.

[38] Peter J. Kostelec and Daniel N. Rockmore. FFTs on the Rotation Group. *Journal of Fourier Analysis and Applications*, 14(2):145–179, 2008.

[39] Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical Message Passing for 3D Graph Networks. *arXiv*, 2102.05013, 2021.

[40] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2018.

[41] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards Understanding Regularization in Batch Normalization. In *ICLR*, 2019.

[42] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. In *NeurIPS*, 2019.

[43] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the Universality of Invariant Networks. In *ICML*, 2019.

[44] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On Learning Sets of Symmetric Elements. In *ICML*, 2020.

[45] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI*, 2019.

[46] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *NeurIPS*, 2020.

[47] Cheol Woo Park, Mordechai Kornbluth, Jonathan Vandermause, Chris Wolverton, Boris Kozinsky, and Jonathan P. Mailoa. Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture. *npj Computational Materials*, 7(1):1–9, 2021.

[48] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R. Manby, and Thomas F. Miller. OrbNet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of Chemical Physics*, 153(12):124111, 2020.

[49] Zhuoran Qiao, Anders S. Christensen, Matthew Welborn, Frederick R. Manby, Anima Anandkumar, and Thomas F. Miller III. UNiTE: Unitary N-body Tensor Equivariant Network with Applications to Quantum Chemistry. *arXiv*, 2105.14655, 2021.

[50] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *ICLR*, 2018.

[51] J. J. Sakurai and San Fu Tuan. *Modern Quantum Mechanics*. Revised, subsequent edition edition, 1993.

[52] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How does batch normalization help optimization? In *NeurIPS*, 2018.

[53] 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 *NeurIPS*, 2017.

[54] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*, 2021.

[55] Muhammed Shuaibi, Adeesh Kolluru, Abhishek Das, Aditya Grover, Anuroop Sriram, Zachary Ulissi, and C. Lawrence Zitnick. Rotation Invariant Graph Neural Networks using Spin Convolutions. *arXiv*, 2106.09575, 2021.

[56] Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds. *arXiv*, 1802.08219, 2018.

[57] 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 *NeurIPS*, 2021.

[58] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. In *NeurIPS*, 2018.

[59] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *CVPR*, 2019.

[60] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019.

[61] Dmitry Yarotsky. Universal Approximations of Invariant Maps by Neural Networks. *Constructive Approximation*, 2021.

A Proof of Theorem 2

We prove the universal approximation theorem by showing the equivalence of TFN and our model. Complex spherical harmonics are related to Clebsch-Gordan coefficients via [51, 3.7.72]

$$Y_{m_i}^{(l_i)}(\hat{\boldsymbol{r}})Y_{m_f}^{(l_f)}(\hat{\boldsymbol{r}}) = \sum_{l_o,m_o} \sqrt{\frac{(2l_i+1)(2l_f+1)}{4\pi(2l_o+1)}} C_{(l_f,0),(l_i,0)}^{(l_o,0)} C_{(l_f,m_f),(l_i,m_i)}^{(l_o,m_o)} Y_{m_o}^{(l_o)}(\hat{\boldsymbol{r}}). \tag{12}$$

We now use the fact that multiplying a learnable function with a unitary matrix or a scalar does not change the resulting function space. We can therefore adapt Eq. (2) by substituting

$$C_{(l_f,m_f),(l_i,m_i)}^{(l_o,m_o)} \mapsto C(l_f,m_f,l_i,m_i,l_o,m_o) = \sqrt{\frac{(2l_i+1)(2l_f+1)}{4\pi(2l_o+1)}} C_{(l_f,0),(l_i,0)}^{(l_o,0)} C_{(l_f,m_f),(l_i,m_i)}^{(l_o,m_o)} \tag{13}$$

without impacting model expressivity. Since real spherical harmonics and complex (conjugate) spherical harmonics cover the same function space, we can furthermore substitute the filter with $F_m'^{(l)}(\boldsymbol{x}) = R^{(l)}(x)Y_m^{(l)*}(\hat{\boldsymbol{x}})$. Using the spherical harmonics expansion we therefore obtain

$$\begin{aligned}
\tilde{\boldsymbol{H}}_a'(\boldsymbol{X},\boldsymbol{H}')(\hat{\boldsymbol{r}}) &= \sum_{l_o,m_o} \tilde{\boldsymbol{H}}_{am_o}'^{(l_o)}(\boldsymbol{X},\boldsymbol{H}')Y_{m_o}^{(l_o)}(\hat{\boldsymbol{r}}) \\
&= \sum_{l_o,m_o} \left(\theta \boldsymbol{H}_{am_o}'^{(l_o)} + \sum_{l_f,m_f} \sum_{l_i,m_i} C(l_f,m_f,l_i,m_i,l_o,m_o) \sum_{b\in\mathcal{N}_a} F_{m_f}'^{(l_f)}(\boldsymbol{x}_{ba}) \boldsymbol{H}_{bm_i}'^{(l_i)} \right) Y_{m_o}^{(l_o)}(\hat{\boldsymbol{r}}) \\
&= \theta \boldsymbol{H}_a'(\hat{\boldsymbol{r}}) + \sum_{l_f,m_f} \sum_{l_i,m_i} \sum_{b\in\mathcal{N}_a} F_{m_f}'^{(l_f)}(\boldsymbol{x}_{ba}) Y_{m_f}^{(l_f)}(\hat{\boldsymbol{r}}) \boldsymbol{H}_{bm_i}'^{(l_i)} Y_{m_i}^{(l_i)}(\hat{\boldsymbol{r}}) \\
&= \theta \boldsymbol{H}_a'(\hat{\boldsymbol{r}}) + \sum_{b\in\mathcal{N}_a} \left(\sum_{l_f,m_f} F_{m_f}'^{(l_f)}(\boldsymbol{x}_{ba}) Y_{m_f}^{(l_f)}(\hat{\boldsymbol{r}}) \right) \left(\sum_{l_i,m_i} \boldsymbol{H}_{bm_i}'^{(l_i)} Y_{m_i}^{(l_i)}(\hat{\boldsymbol{r}}) \right) \\
&= \theta \boldsymbol{H}_a'(\hat{\boldsymbol{r}}) + \sum_{b\in\mathcal{N}_a} F'(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}}) \boldsymbol{H}_b'(\hat{\boldsymbol{r}}).
\end{aligned} \tag{14}$$

These functions rely on complex-valued representations, while the output and $\mathrm{SO}(3)$ representations are real-valued. However, we can restrict the representations to real values without changing the resulting function space. To see this, we look at the result's real component

$$\begin{aligned}
\Re[\tilde{\boldsymbol{H}}_a'(\boldsymbol{X},\boldsymbol{H}')(\hat{\boldsymbol{r}})] &= \theta \Re[\boldsymbol{H}_a'(\hat{\boldsymbol{r}})] + \sum_{b\in\mathcal{N}_a} \Re[F'(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}}) \boldsymbol{H}_b'(\hat{\boldsymbol{r}})] \\
&= \theta \Re[\boldsymbol{H}_a'(\hat{\boldsymbol{r}})] + \sum_{b\in\mathcal{N}_a} (\Re[F'(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}})]\Re[\boldsymbol{H}_b'(\hat{\boldsymbol{r}})] - \Im[F'(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}})]\Im[\boldsymbol{H}_b'(\hat{\boldsymbol{r}})]).
\end{aligned} \tag{15}$$

The function space covered by $\Re[F'(\boldsymbol{x},\hat{\boldsymbol{r}})]$, and thus $\Re[\boldsymbol{H}'(\hat{\boldsymbol{r}})]$, is the same as $\Im[F'(\boldsymbol{x},\hat{\boldsymbol{r}})]$, and thus $\Im[\boldsymbol{H}'(\hat{\boldsymbol{r}})]$. We can therefore simply remove the imaginary part without changing the resulting function space, obtaining

$$\begin{aligned}
\tilde{\boldsymbol{H}}_a^{\text{sphere}}(\boldsymbol{X},\boldsymbol{H})(\hat{\boldsymbol{r}}) &= \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}) + \sum_{b\in\mathcal{N}_a} \Re[F'(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}})] \boldsymbol{H}_b(\hat{\boldsymbol{r}}) \\
&= \theta \boldsymbol{H}_a(\hat{\boldsymbol{r}}) + \sum_{b\in\mathcal{N}_a} F_{\text{sphere}}(\boldsymbol{x}_{ba},\hat{\boldsymbol{r}}) \boldsymbol{H}_b(\hat{\boldsymbol{r}}).
\end{aligned} \tag{16}$$

$\mathcal{F}_{\text{feat}}^{\text{sphere}}(D)$ thus spans the exact same space of embedding functions as $\mathcal{F}_{\text{feat}}^{\text{TFN}}(D)$, despite only using real functions on the S^2 sphere. However, we cannot span the full space of rotationally equivariant linear pooling functions, since equivariant linear functions on the S^2 sphere are limited to convolutions with zonal filters [21]. Fortunately, scalar pooling functions are limited to linear functions of the constant $l = 0$ part. This is equivalent to integrating over the real-space spherical representation, as done in $\mathcal{F}_{\text{pool}}^{\text{sphere}}$. □

B Proof of Theorem 3

To prove this theorem we first introduce a proposition by Villar et al. [57].

Proposition A (Villar et al. [57])**.** *If h is an $\mathrm{SO}(d)$-equivariant function $\mathbb{R}^{d\times n} \to \mathbb{R}^d$ of n vector inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, then there are n $\mathrm{SO}(d)$-invariant functions $f_c\colon \mathbb{R}^{d\times n} \to \mathbb{R}$ such that*

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sum_{c=1}^{n} f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)\boldsymbol{x}_c, \tag{17}$$

except when $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ span a $(d-1)$-dimensional space. In that case, there exist $\mathrm{O}(d)$-invariant functions $f_c\colon \mathbb{R}^{d\times n} \to \mathbb{R}$ such that

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sum_{c=1}^{n} f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)\boldsymbol{x}_c + \sum_{S\in\binom{[n]}{d-1}} f^{(S)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)\boldsymbol{x}_S, \tag{18}$$

where $[n] := \{1, \ldots, n\}$, $\binom{[n]}{d-1}$ is the set of all $(d-1)$-subsets of $[n]$, and $\boldsymbol{x}_S$ is the generalized cross product of vectors $\boldsymbol{x}_i$ with $i \in S$ (taken in ascending order).

To extend Prop. A to our case, we need to restrict the functions to being translation-invariant and permutation-equivariant. We will only concern ourselves with the case where the vectors do not span a $(d-1)$-dimensional space. We start by considering translation-invariant functions, following the proof idea of Villar et al. [57, Lemma 7].

Lemma A. *Let h be a translation-invariant and $\mathrm{SO}(d)$-equivariant function $\mathbb{R}^{d\times n} \to \mathbb{R}^d$ of n vector inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. Let $\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1$ not span a $(d-1)$-dimensional space. Then there are $n-1$ translation- and $\mathrm{SO}(d)$-invariant functions $f_c\colon \mathbb{R}^{d\times n} \to \mathbb{R}$ such that*

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sum_{c=2}^{n} f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)(\boldsymbol{x}_c - \boldsymbol{x}_1). \tag{19}$$

Proof. Consider the $\mathrm{SO}(d)$-equivariant function $\tilde{h}\colon \mathbb{R}^{d\times(n-1)} \to \mathbb{R}^d$ with

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = h(0, \boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1) = \tilde{h}(\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1). \tag{20}$$

Due to Prop. A we have

$$\tilde{h}(\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1) = \sum_{c=2}^{n} \tilde{f}^{(c)}(\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1)(\boldsymbol{x}_c - \boldsymbol{x}_1), \tag{21}$$

with the $\mathrm{SO}(d)$-equivariant function $\tilde{f}^{(c)}$. If we now substitute $\tilde{f}^{(c)}$ with the $\mathrm{SO}(d)$-equivariant and translation-invariant function $f^{(c)}$, i.e.

$$\tilde{f}^{(c)}(\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1) = f^{(c)}(0, \boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1) = f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n), \tag{22}$$

we obtain

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sum_{c=2}^{n} f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)(\boldsymbol{x}_c - \boldsymbol{x}_1). \tag{23}$$

□

Next, we extend this result to permutation-equivariant functions.

Lemma B. *Let h be a translation-invariant, and permutation and $\mathrm{SO}(d)$-equivariant function $\mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ of n vector inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. Let $\boldsymbol{x}_2 - \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{x}_1$ not span a $(d-1)$-dimensional space. Then there are $n-1$ translation- and $\mathrm{SO}(d)$-invariant, and permutation-equivariant functions $f_c\colon \mathbb{R}^{d\times n} \to \mathbb{R}^n$ such that*

$$h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sum_{c=2}^{n} f^{(c)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)(\boldsymbol{x}_c - \boldsymbol{x}_1). \tag{24}$$

Proof. Permutation equivariance implies that for all s and t (w.l.o.g. $s < t$)

$$h_s(\ldots, \boldsymbol{x}_s, \ldots, \boldsymbol{x}_t, \ldots) = h_t(\ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_s, \ldots). \tag{25}$$

Due to Lemma A we have

$$h_s(\ldots, \boldsymbol{x}_s, \ldots, \boldsymbol{x}_t, \ldots) = \sum_{c=2}^{n} f_s^{(c)}(\ldots, \boldsymbol{x}_s, \ldots, \boldsymbol{x}_t, \ldots)(\boldsymbol{x}_c - \boldsymbol{x}_1), \tag{26}$$

$$= h_t(\ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_s, \ldots) = \sum_{c=2}^{n} f_t^{(c)}(\ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_s, \ldots)(\boldsymbol{x}_c - \boldsymbol{x}_1), \tag{27}$$

with $n-1$ SO(d)- and translation-invariant functions $f^{(c)} \colon \mathbb{R}^{d\times n} \to \mathbb{R}^n$. We can solve this equation by choosing

$$f_s^{(c)}(\ldots, \boldsymbol{x}_s, \ldots, \boldsymbol{x}_t, \ldots) = f_t^{(c)}(\ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_s, \ldots), \tag{28}$$

i.e. permutation-equivariant functions $f^{(c)}$. □

Finally, to bring Lemma B to the form presented in the theorem, we first observe that adding scalar inputs $\boldsymbol{H}$ does not affect the proofs in this section. Second, we observe that subtracting by $\boldsymbol{x}_1$ in Eq. (20) is arbitrary. To bring this more in line with GNNs we can instead subtract the input of each h_a by $\boldsymbol{x}_a$. This yields

$$h_a(\boldsymbol{X}, \boldsymbol{H}) = \sum_{\substack{c=1\\ c\neq a}}^{n} f_a^{(c)}(\boldsymbol{X}, \boldsymbol{H})(\boldsymbol{x}_c - \boldsymbol{x}_a). \tag{29}$$

□

C Proof of Lemma 1

Using the fact that the Wigner-D matrix is unitary, we obtain for any rotation matrix $\boldsymbol{R}$:

$$\begin{aligned}
F_{\text{sphere}}(\boldsymbol{R}\boldsymbol{x}, \boldsymbol{R}\hat{\boldsymbol{r}}) &= \sum_{l,m} R^{(l)}(x) \Re[Y_m^{(l)*}(\boldsymbol{R}\hat{\boldsymbol{x}}) Y_m^{(l)}(\boldsymbol{R}\hat{\boldsymbol{r}})] \\
&= \sum_{l,m,m',m''} R^{(l)}(x) \Re[Y_{m'}^{(l)*}(\hat{\boldsymbol{x}}) D_{m,m'}^{(l)*}(\boldsymbol{R}) D_{m,m''}^{(l)}(\boldsymbol{R}) Y_{m''}^{(l)}(\hat{\boldsymbol{r}})] \\
&= \sum_{l,m',m''} R^{(l)}(x) \Re[Y_{m'}^{(l)*}(\hat{\boldsymbol{x}}) \delta_{m',m''} Y_{m''}^{(l)}(\hat{\boldsymbol{r}})] \\
&= \sum_{l,m'} R^{(l)}(x) \Re[Y_{m'}^{(l)*}(\hat{\boldsymbol{x}}) Y_{m'}^{(l)}(\hat{\boldsymbol{r}})] = F_{\text{sphere}}(\boldsymbol{x}, \hat{\boldsymbol{r}}).
\end{aligned} \tag{30}$$

□

D Efficient message passing

For clarity we demonstrate how to optimize the summation order using the simpler one-hop message passing. For a regular Hadamard product we reorder the sums as

$$\begin{aligned}
\boldsymbol{m}_{(ca)i} &= \sum_{b\in\mathcal{N}_a\setminus\{c\}} \Big(\boldsymbol{W}^{(2)}\boldsymbol{W}^{(1)}\boldsymbol{e}_{\text{CBF}}(x_{ca}, \varphi_{bac})\Big)_i \boldsymbol{m}_{(ba)i} \\
&= \sum_{b\in\mathcal{N}_a\setminus\{c\}} \Big(\sum_j \sum_l \sum_n \boldsymbol{W}_{ij}^{(2)} \boldsymbol{W}_{j(ln)}^{(1)} \boldsymbol{e}_{\text{CBF}}^{\text{rad}}(x_{ca})_{ln} \boldsymbol{e}_{\text{CBF}}^{\text{SH}}(\varphi_{bac})_l\Big) \boldsymbol{m}_{(ba)i} \\
&= \sum_j \boldsymbol{W}_{ij}^{(2)} \sum_l \Big(\sum_n \boldsymbol{W}_{j(ln)}^{(1)} \boldsymbol{e}_{\text{CBF}}^{\text{rad}}(x_{ca})_{ln}\Big)\Big(\sum_{b\in\mathcal{N}_a\setminus\{c\}} \boldsymbol{e}_{\text{CBF}}^{\text{SH}}(\varphi_{bac})_l \boldsymbol{m}_{(ba)i}\Big).
\end{aligned} \tag{31}$$

For a bilinear layer we use

$$\begin{aligned}
\boldsymbol{m}_{(ca)i} &= \sum_{b \in \mathcal{N}_a \setminus \{c\}} \left(\left(\boldsymbol{W}^{(1)} \boldsymbol{e}_{\text{CBF}}(x_{ca}, \varphi_{bac}) \right)^T \mathbf{W}^{(2)} \boldsymbol{m}_{(ba)} \right)_i \\
&= \sum_{b \in \mathcal{N}_a \setminus \{c\}} \sum_{i'} \sum_{j} \left(\sum_{l} \sum_{n} \boldsymbol{W}^{(1)}_{j(ln)} \boldsymbol{e}^{\text{rad}}_{\text{CBF}}(x_{ca})_{ln} \boldsymbol{e}^{\text{SH}}_{\text{CBF}}(\varphi_{bac})_l \right) \mathbf{W}^{(2)}_{iji'} \boldsymbol{m}_{(ba)i'} \\
&= \sum_{j} \sum_{i'} \mathbf{W}^{(2)}_{iji'} \sum_{l} \left(\sum_{n} \boldsymbol{W}^{(1)}_{j(ln)} \boldsymbol{e}^{\text{rad}}_{\text{CBF}}(x_{ca})_{ln} \right) \left(\sum_{b \in \mathcal{N}_a \setminus \{c\}} \boldsymbol{e}^{\text{SH}}_{\text{CBF}}(\varphi_{bac})_l \boldsymbol{m}_{(ba)i'} \right).
\end{aligned} \tag{32}$$

Note that since $\boldsymbol{W}^{(1)}$ is shared across layers we only need to calculate the sum over n once.

E Variance after message passing

The layer-wise variance after sum aggregation is

$$\text{Var}_i \Big[\sum_{b \in \mathcal{N}_a} \boldsymbol{m}_{(ba)i} \Big] = \sum_{b \in \mathcal{N}_a} \text{Var}_i[\boldsymbol{m}_{(ba)i}] + \sum_{b \in \mathcal{N}_a} \sum_{c \in \mathcal{N}_a \setminus \{b\}} \text{Cov}_i[\boldsymbol{m}_{(ba)i}, \boldsymbol{m}_{(ca)i}]. \tag{33}$$

This variance depends on the number of neighbors in $\mathcal{N}_a$. However, we consistently found that rescaling the output depending on $\mathcal{N}_a$ has negative effects on the accuracy. The likely reason for this is that atomic interactions scale roughly linearly with neighborhood size. Moreover, the covariance in Eq. (33) is not zero since all messages $\boldsymbol{m}_{ba}$ are transformed using the same weight matrices. We therefore best estimate this variance empirically.

For a Hadamard product-based message passing filter (and analogously for a bilinear layer) we have

$$\text{Var}_i[F_i \boldsymbol{m}_i] = \text{Cov}_i[F_i^2, \boldsymbol{m}_i^2] + (\text{Var}_i[F_i] + \mathbb{E}_i[F_i]^2)(\text{Var}_i[\boldsymbol{m}_i] + \mathbb{E}_i[\boldsymbol{m}_i]^2) - (\text{Cov}_i[F_i, \boldsymbol{m}_i] + \mathbb{E}_i[F_i]\mathbb{E}_i[\boldsymbol{m}_i])^2. \tag{34}$$

The main problem with this covariance is the non-zero quadratic covariance $\text{Cov}_i[F_i^2, \boldsymbol{m}_i^2]$. We again estimate this variance empirically based on a data sample.

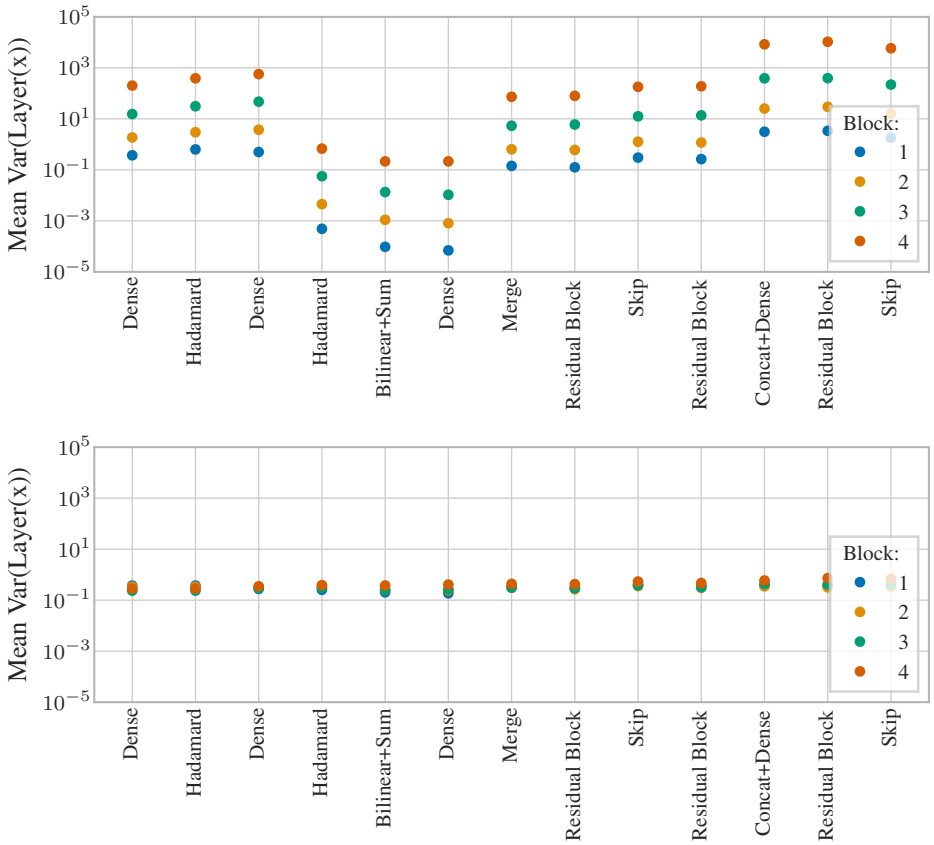

Figure 3: Layer-wise activation variance. GemNet's variance varies strongly between layers and increases significantly after each block without scaling factors (top). Introducing scaling factors successfully stabilizes the variance (bottom).

F GemNet architecture

We use 4 stacked interaction blocks and an embedding size of 128 throughout the model. For the basis functions we choose $N_{\text{SHBF}} = N_{\text{CHBF}} = 7$ and $N_{\text{SRBF}} = N_{\text{CRBF}} = N_{\text{RBF}} = 6$. For the weight tensor of the bilinear layer in the interaction block we use $N_{\text{bilinear,SBF}} = 32$ and $N_{\text{bilinear,CBF}} = 64$. We found that sharing the first weight matrix in Eq. (11), the down projection, resulted in the same validation loss but reduced the training time by up to 15 %. The down projection size was chosen as 16 for the radial and circular basis and 32 for the spherical basis.

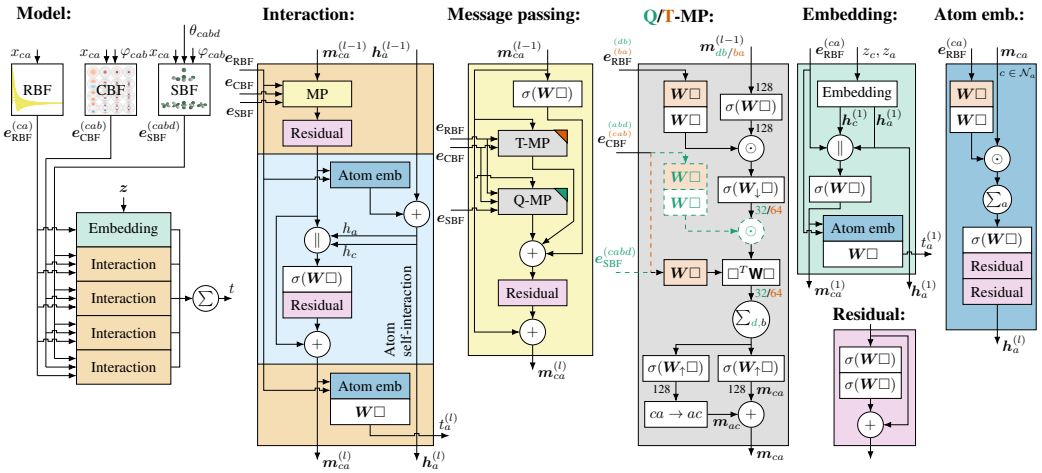

Figure 4: The full GemNet architecture. $\square$ denotes the layer's input, $\|$ concatenation, σ a non-linearity (we use SiLU in this work [20]), and orange a layer with weights shared across interaction blocks. Differences between two-hop message passing (Q-MP) and one-hop message passing (T-MP) are denoted by dashed lines. Numbers next to connecting lines denote embedding sizes.

G Training and hyperparameters

Table 6: Model and training hyperparameters.

Hyperparameters			
Interaction cutoff c_{int}		10 Å	
Embedding cutoff c_{emb}		5 Å	
Learning rate		1×10^{-3}	
EMA decay		0.999	
Weight decay		2×10^{-6}	
Decay epochs		1200	
Decay rate		0.01	
Decay factor on plateau		0.5	
Gradient clipping threshold		10.0	
Envelope exponent		5	
Force weighting factor ρ		0.999	
	MD17	MD17@CCSD(T)	Coll
Train set size	1000	950	120 000
Val. set size	1000	50	10 000
Max epochs	2000	2000	400
Evaluation interval (epochs)	10	10	2
Decay on plateau patience (epochs)	50	50	10
Decay on plateau cooldown (epochs)	50	50	10
Warm-up epochs	10	10	1
Batch size	1	1	32

We subtract the mean energy from each molecule in MD17 to obtain a training target similar to atomization energy. We train on eV for energies and eV Å^{-1} for forces. As a training objective we use the weighted loss function

$$\mathcal{L}_{\text{MD}}(\boldsymbol{X},\boldsymbol{z}) = (1-\rho)\left|f_\theta(\boldsymbol{X},\boldsymbol{z}) - \hat{t}(\boldsymbol{X},\boldsymbol{z})\right| + \frac{\rho}{N}\sum_{i=1}^{N}\sqrt{\sum_{\alpha=1}^{3}\left(-\frac{\partial f_\theta(\boldsymbol{X},\boldsymbol{z})}{\partial \boldsymbol{x}_{i\alpha}} - \hat{F}_{i\alpha}(\boldsymbol{X},\boldsymbol{z})\right)^2}, \tag{35}$$

with force weighting factor $\rho = 0.999$. We found the selection of the batch size to be of great influence on the model's performance for the MD17(@CCSD) dataset. Changing the batch size from

32 to 1 resulted in an approx. 25 % lower validation MAE. The learning rate of 1×10^{-3} and the selection of the embedding cutoff $c_{emb} = 5$ Å and interaction cutoff $c_{int} = 10$ Å are rather important hyperparameters as well, see Table 8. We optimized the model using AMSGrad [50] with weight decay [40] in combination with a linear learning rate warm-up, exponential decay and decay on plateau. However, we did not apply the weight decay for the initial atom embeddings, biases and frequencies (used in the radial basis). Without weight decay the force MAE was around 3 % higher on COLL (not on OC20). Gradient clipping and early stopping on the validation loss were used as well. In addition, we divided the gradients of weights that are shared across multiple blocks by the number of blocks the weights are shared for, which resulted in a small gain in accuracy. The model weights for validation and test were obtained using an exponential moving average (EMA) with decay rate 0.999. The used hyperparameters can be found in Table 6. The combined model on revised MD17 was trained with a batch size of 10.

We used a slightly adapted model for the OC20 dataset. It uses 128 Gaussian radial basis functions instead of spherical Bessel functions, which do not depend on the degree l of the spherical harmonic. We furthermore used only three interaction blocks, an atom and edge embedding size of 512, an embedding cutoff of 6 Å, a learning rate of $5{\times}10^{-4}$, no weight decay, only learning rate decay on plateau with a patience of 15 000 steps and a factor of 0.8 (no warm-up or exponential decay), and a batch size of 2048.

H Additional experimental results

Table 7: MAE for direct force predictions on MD17 in meV/Å. The increased speed of direct force predictions comes at a significant cost of accuracy. Note that the direct models are still more accurate than many previous models.

	GemNet-Q	GemNet-T	GemNet-dQ	GemNet-dT
Aspirin	9.4	9.5	17.8	18.0
Benzene[9]	6.3	6.3	8.5	8.0
Benzene[10]	1.5	1.4	2.5	2.3
Ethanol	3.8	3.7	6.4	6.8
Malonaldehyde	6.9	6.7	11.5	12.5
Naphthalene	2.2	2.4	5.2	5.9
Salicylic acid	5.4	5.5	12.9	13.2
Toluene	2.6	2.6	6.1	5.7
Uracil	4.5	4.2	11.7	10.9

Table 8: Impact of the cutoff on force MAE on COLL. Results reported in meV/Å after 500 000 training steps. Increasing the interaction cutoff to 10 Å slightly reduces the error. Decreasing the embedding cutoff to 3 Å significantly increases the error.

c_{emb}/Å	c_{int}/Å	MAE
5	10	27.0
5	5	28.2
3	10	33.4
3	5	35.3

Table 9: Force MAE for MD17 in meV/Å. GemNet using 1000 training samples compared to SchNet using 50 000 samples. GemNet outperforms SchNet on six out of eight molecules – despite using 50x fewer samples.

	SchNet 50k	GemNet-Q	GemNet-T
Aspirin	14.3	**9.4**	9.5
Benzene[9]	7.4	**6.3**	**6.3**
Benzene[10]	-	1.5	**1.4**
Ethanol	**2.2**	3.8	3.7
Malonaldehyde	**3.5**	6.9	6.7
Naphthalene	4.8	**2.2**	2.4
Salicylic acid	8.2	**5.4**	5.5
Toluene	3.9	**2.6**	**2.6**
Uracil	4.8	4.5	**4.2**

Table 10: Effect of adding our independent improvements to DimeNet^{++} on force MAE for COLL in meV/Å. In this experiment we increased the basis embedding size of DimeNet^{++} from 8 to 16 to eliminate this bottleneck. All improvements have a significant effect.

Model	Forces
DimeNet^{++}	41.1
with symmetric message passing	37.5
with bilinear layer	38.6
with scaling factors	40.0

Table 11: Force MAE for the revised MD17 dataset [11] in meV/Å. On average, GemNet outperforms FCHL19 by 52 % and even UNiTE by 5 %, which is a Δ-ML approach based on quantum mechanical features [49].

	FCHL19	UNiTE	GemNet-Q	GemNet-T
Aspirin	20.9	**7.8**	9.7	9.5
Benzene	2.6	0.7	0.7	**0.5**
Ethanol	6.2	4.2	**3.6**	**3.6**
Malonaldehyde	10.3	7.1	6.7	**6.6**
Naphthalene	6.5	2.4	**1.9**	2.1
Salicylic acid	9.5	**4.1**	5.3	5.5
Toluene	8.8	2.9	2.3	**2.2**
Uracil	4.2	**3.8**	4.1	**3.8**

Table 12: Force MAE of different models (number of parameters in parentheses) for the MD17 dataset in meV/Å. GemNet performs worse with an embedding size of 64, but still substantially better than previous models with more parameters.

	PaiNN (600k)	DimeNet (1.9M)	GemNet-T 64 (490k)	GemNet-T (1.9M)
Aspirin	14.7	21.6	11.2	**9.5**
Benzene[9]	-	8.1	-	**6.3**
Benzene[10]	-	-	**1.1**	1.4
Ethanol	9.7	10.0	5.1	**3.7**
Malonaldehyde	14.9	16.6	7.8	**6.7**
Naphthalene	3.3	9.3	3.3	**2.4**
Salicylic acid	8.5	16.2	6.9	**5.5**
Toluene	4.1	9.4	3.3	**2.6**
Uracil	6.0	13.1	5.3	**4.2**

Table 13: Force MAE of GemNet on the revised MD17 dataset [11] in meV/Å when using individual models for each molecule ("Individual") versus a single model for all molecules ("Combined"). The combined setting is harder to learn, leading to a higher error in most cases. GemNet-Q performs better than GemNet-T in this setting.

	GemNet-Q		GemNet-T	
	Individual	Combined	Individual	Combined
Aspirin	9.7	10.0	9.5	9.9
Benzene	0.7	0.5	0.5	0.6
Ethanol	3.6	4.4	3.6	4.9
Malonaldehyde	6.7	7.7	6.6	8.3
Naphthalene	1.9	1.9	2.1	2.2
Salicylic acid	5.3	4.6	5.5	5.0
Toluene	2.3	2.2	2.2	2.5
Uracil	4.1	4.1	3.8	4.3

I Computation time

The models were trained primarily using Nvidia GeForce GTX 1080Ti GPUs. For MD17 and MD17@CCSD training the direct force prediction variants took less than two days, GemNet-Q and GemNet-T took around 6 days per molecule but with very little progress after the 100 hour mark. However, thanks to the memory efficient implementation and the low batch size used, several models were trained in parallel on a single GPU. On the COLL dataset training the direct force prediction variants took around 24 hours each. GemNet-T trained for 60 hours, while GemNet-Q took 6 days. However, after 60 hours GemNet-Q is already within 5 % of its final validation error and outperforms GemNet-T by a large margin. Note that the training time reduces dramatically when using a larger batch size, at the cost of a slightly higher MAE on MD17.

Table 14: Runtime per batch of Toluene molecules on an Nvidia GeForce GTX 1080Ti in seconds. GemNet-T is comparably fast to previous methods. Note that NequIP requires roughly 10x more training epochs than GemNet for convergence [4]. Using direct force predictions and only one-hop message passing significantly accelerates training and inference (GemNet-dT). Efficient aggregation allows for the usage of a bilinear layer instead of a Hadamard product at no additional cost (GemNet-Q vs. Hadamard-Eff) and enables training with higher batch sizes (Hadamard-Eff vs. Hadamard-NonEff). Note that our implementation does not focus on runtime and can likely be significantly optimized.

	batch size 32		batch size 4	
	Training	Inference	Training	Inference
DimeNet^{++}	0.357	0.065	0.283	0.031
NequIP (l=1)	0.066	0.042	0.070	0.044
NequIP (l=3, reflections)	0.336	0.206	0.327	0.197
GemNet-Q	1.067	0.376	0.628	0.099
GemNet-T	0.397	0.088	0.299	0.038
GemNet-dQ	0.369	0.264	0.106	0.052
GemNet-dT	0.134	0.067	0.065	0.020
Hadamard-Eff	1.077	0.392	0.632	0.103
Hadamard-NonEff	OOM	0.378	0.633	0.103