Implicit Neural Convolutional Kernels for Steerable CNNs

Maksim Zhdanov
Helmholtz-Zentrum
Dresden-Rossendorf
maxxxzdn@gmail.com

Nico Hoffmann
Helmholtz-Zentrum
Dresden-Rossendorf
n.hoffmann@hzdr.de

Gabriele Cesa
Qualcomm AI Research
University of Amsterdam
cesa.gabriele@gmail.com

Abstract

Steerable convolutional neural networks (CNNs) provide a general framework for building neural networks equivariant to translations and other transformations belonging to an origin-preserving group G, such as reflections and rotations. They rely on standard convolutions with G-steerable kernels obtained by analytically solving the group-specific equivariance constraint imposed onto the kernel space. As the solution is tailored to a particular group G, the implementation of a kernel basis does not generalize to other symmetry transformations, which complicates the development of group equivariant models. We propose using implicit neural representation via multi-layer perceptrons (MLPs) to parameterize G-steerable kernels. The resulting framework offers a simple and flexible way to implement Steerable CNNs and generalizes to any group G for which a G-equivariant MLP can be built. We apply our method to point cloud (ModelNet-40) and molecular data (QM9) and demonstrate a significant improvement in performance compared to standard Steerable CNNs.

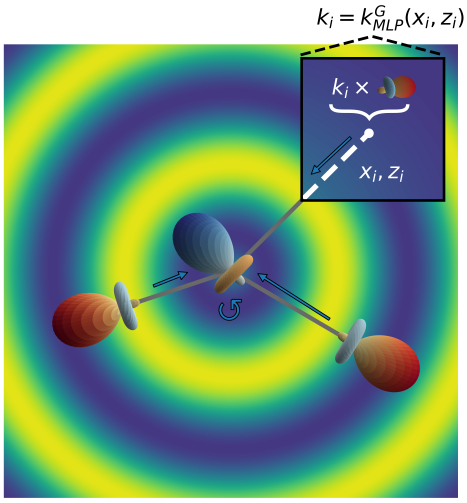

Figure 1. Illustration of the proposed approach: computing the response of an implicit kernel k (background) of a steerable point convolution for an arbitrary atom i of a molecular graph. Each atom is endowed with spatial information x, attributes z and steerable features, which we visualize as spherical functions.

1. Introduction

Equivariant deep learning methods have proven to be an effective toolkit for high-dimensional learning tasks when the symmetry[1] of the corresponding data domain is known. Incorporating the knowledge into a neural network model in the form of inductive biases allows one to significantly restrict the hypothesis class of functions and thus improve the data efficiency of a model as well as its generalization performance [6]. A well-known example would be convolutional neural networks [22] (CNNs) that are *equivariant* to translations, meaning that if the input image is translated, the resulting intermediate feature maps are also translated accordingly. Recently, CNNs were generalized to exploit a larger number of symmetries via group convolutions yielding a class of group-equivariant CNNs (G-CNNs) [9]. As mentioned in the name, such models are equivariant to the desired symmetry group, which might encompass sets of transformations broader than translations. A prominent example would be the Euclidean group $E(n)$, which comprises translations, rotations and reflections in n-dimensional Euclidean space and underlies many important problems in physics and chemistry, such as molecular modelling or point clouds. Specifically, there are properties of objects defined in physical space that are invariant to Euclidean transformations (e.g. rotating furniture in space does not change its label) or equivariant to them (e.g. atom velocities in a molecule must be rotated correspondingly to the rotation of the coordinate system). Furthermore, it is often desired to respect the underlying symmetry in a system to make the model perform according to our expectations.

There are multiple ways to parameterize neural networks

[1] Informally, symmetry means a transformation of an object that preserves certain properties of the object.

to incorporate equivariance to the Euclidean group. One possible choice is to employ a message-passing neural network as a backbone and compute/update messages equivariantly via convolutions. This approach enjoys high expressivity of graph neural networks and generalizes well to point clouds and graphs. Depending on the convolutional domain, one can further categorize equivariant convolutional operators into regular [3, 9, 12, 20] and *steerable* group convolutions [10, 11, 41]. The latter are considered particularly suitable for incorporating physical quantities such as forces or velocity into learned functions, which is desirable as leveraging the information leads to a generally improved performance [5]. The key idea behind Steerable CNNs is the use of standard convolution - which guarantees translation equivariance - with G-steerable kernels that ensure commutativity with the transformations of another group G, such as rotations. The commutation requirement imposes a constraint onto the kernel space that must be solved analytically for each group G. This, in turn, does not allow generalizing a convolution operator tailored to a specific group to other symmetry transformations. In the case of the Euclidean group, Cesa *et al.* [7] proposed a generally applicable way of parameterizing steerable convolutions for sub-groups of $E(n)$. The method relies on adapting a pre-defined kernel basis explicitly developed for the group $E(n)$ to an arbitrary sub-group by using *group restriction*.

However, because only a finite basis can be chosen, a basis tailored for $E(n)$ can be sub-optimal in terms of expressiveness for its sub-groups; see Section 3.2 more details. Hence, we propose an alternative way of building steerable convolutions based on implicit neural kernels, i.e. convolutional kernels implemented as continuous functions parameterized by MLPs [29, 30]. We demonstrate how G-steerable convolutions with implicit neural kernels can be implemented from scratch for any sub-group G of the orthogonal group $O(n)$. The method allows us to ultimately minimize requirements to implement equivariance to new groups; see Section 3.4. The flexibility of neural functions also permits the injection of geometric and physical quantities in point convolutions, increasing their expressiveness. We validate our framework on point-cloud data (ModelNet-40 [45]) and molecular data (QM9 [44]) and indicate that implicit neural kernels improve the overall performance of steerable group convolutions.

2. Background: Steerable Convolutions

In this work, we propose a general solution to easily build Steerable CNNs equivariant to translations and any compact group G. In Section 2.1, we provide some necessary prerequisites from group theory and group representation theory [34]. Then, we review the framework of Steerable CNNs and discuss the constraint it induces on the convolutional kernels in Section 2.2.

2.1. Groups, Representations and Equivariance

Definition 1 (Group)**.** *A group is an algebraic structure that consists of a set G and a binary operator $\circ : G \times G \to G$ called the group product (denoted by juxtaposition for brevity $g \circ h = gh$) that satisfies the following axioms:*

- *Closure:* $\forall g, h \in G : gh \in G$;
- *Identity:* $\exists!\ e \in G : eg = ge = g\ \forall g \in G$;
- *Inverse:* $\forall g \in G\ \exists!\ g^{-1} \in G : gg^{-1} = g^{-1}g = e$;
- *Associativity:* $(gh)k = g(hk)\ \forall g, h, k \in G$.

Definition 2 (Group action)**.** *An action of a group G on a set $\mathcal{X}$ is a mapping $(g, x) \mapsto g.x$ associating a group element $g \in G$ and a point $x \in \mathcal{X}$ with some other point on $\mathcal{X}$ such that the following holds:*

$$g.(h.x) = (gh).x \qquad \forall g, h \in G, x \in \mathcal{X}$$

Definition 3 (Group representation)**.** *A linear representation ρ of a group G is a map $\rho : G \to \mathbb{R}^{d \times d}$ that assigns an invertible matrix $\rho(g)\ \forall g \in G$ and satisfies the following condition*

$$\rho(gh) = \rho(g)\rho(h)\ \forall g, h \in G.$$

A group representation $\rho(g) : V \to V$ furthermore can be seen as a linear action of G on a vector space V. Additionally, if two (or more) vectors $v_1 \in \mathbb{R}^{d_1}$ and $v_2 \in \mathbb{R}^{d_2}$ belong to vector spaces transforming under representations ρ_1 and ρ_2, their concatenation $v_1 \oplus v_2 \in \mathbb{R}^{d_1+d_2}$ transforms under the *direct sum* representation $\rho_1 \oplus \rho_2$. $(\rho_1 \oplus \rho_2)(g)$ is a $d_1 + d_2$ dimensional block-diagonal matrix containing $\rho_1(g)$ and $\rho_2(g)$ in its diagonal blocks.

There are three types of group representations that are important for the definition of our method:

- **Trivial representation:** all elements of G act on V as the identity mapping of V, i.e. $\rho(g) = I$.
- **Standard representation:** the group $O(n)$ of all orthogonal $n \times n$ matrices has a natural action on $V = \mathbb{R}^n$; similarly, if G is a subgroup of $O(n)$, elements of G can act on $V = \mathbb{R}^n$ via the inclusion mapping, i.e. $\rho_{st}(g) = g \in \mathbb{R}^{n \times n}$.
- **Irreducible representations** are a collection of generally known representations that can be used as a building block for larger representations via *direct sum*. As argued in [39], Steerable CNNs can be parameterized w.l.o.g. solely in terms of irreducible representations.

Definition 4 (Equivariance)**.** *Let us have two spaces $\mathcal{X}, \mathcal{Y}$ endowed with a symmetry group G, i.e. with an action defined on them. A function $\phi : \mathcal{X} \to \mathcal{Y}$ is called G-*equivariant*, if it commutes with the action of G on the two spaces:*

$$\phi(g.x) = g.\phi(x) \tag{1}$$

As we have discussed, the layers of conventional CNNs are translation equivariant by design; however, they do not commute with transformations of other groups such as rotations and reflections.

2.2. Steerable CNNs

Steerable CNNs provide a more general framework that allows building convolutions that are equivariant to a group of *isometries* $H \leq E(n)$, decomposable as a semi-direct[2] product of the translations group $(\mathbb{R}^n, +)$ and an origin-preserving compact[3] group $G \leq O(n)$:

$$H = (\mathbb{R}^n, +) \rtimes G \tag{2}$$

where $O(n)$ is the group of n-dimensional rotations and reflections. As translation equivariance is guaranteed [41] by the convolution operator itself, one only has to ensure equivariance to G. See [39–41] for a more in-depth description of Steerable CNNs.

Steerable CNNs operate on *feature fields*. A feature field of type ρ is a feature map $f : \mathbb{R}^n \to \mathbb{R}^d$ endowed with a *group representation* $\rho : G \to \mathbb{R}^{d\times d}$ that defines how an element $g \in G$ transforms the feature:

$$[g.f](x) := \rho(g) f(g^{-1}.x) \tag{3}$$

Typically, each intermediate feature field is associated with its own type. The whole architecture is equivariant when each steerable layer preserves the transformation law of its input and output feature fields. G-equivariant convolution is achieved by applying the following G-specific equivariance constraint onto the space of conventional convolution kernels [41]:

$$k(g.x) = \rho_{out}(g) k(x) \rho_{in}(g)^{-1} \qquad \forall g \in G, x \in \mathbb{R}^n \tag{4}$$

where $k : \mathbb{R}^n \to \mathbb{R}^{d_{out}\times d_{in}}$, and $\rho_{in} : G \to \mathbb{R}^{d_{in}\times d_{in}}$, $\rho_{out} : G \to \mathbb{R}^{d_{out}\times d_{out}}$ are respective representations of input and output feature fields.

To parameterize the kernel k, the constraint in equation 4 needs to be solved *analytically* for each specific group G of interest. This renders a general solution challenging to obtain and limits the applicability of steerable convolutions.

3. Implicit neural kernels

Instead of deriving a steerable kernel basis for each particular group G, we propose parameterizing the kernel $k : \mathbb{R}^n \to \mathbb{R}^{d_{out}\times d_{in}}$ with an MLP satisfying the constraint in Eq. 4. The approach only requires the G-equivariance of the MLP and suggests a flexible framework of implicit steerable convolutions that generalizes to arbitrary groups $G \leq O(n)$. We argue about the minimal requirements of this approach in Section 3.4.

We first define the kernel as an equivariant map between vector spaces that we model with an MLP (see section 3.1). Then we demonstrate that G-equivariance of an MLP is a sufficient condition for building the implicit representation of steerable kernels for compact groups. We compare our method with the solution strategy proposed in [7] in section 3.2. We indicate that the flexibility of neural representation allows expanding the input of a steerable kernel in section 3.3. We describe how a G-equivariant MLP can be implemented in section 3.4. Finally, we describe how one can implement G-steerable point convolution in the form of equivariant message passing [5, 31] in section 3.5.

3.1. Kernel vectorization and equivariance

Our goal is to implement the kernel $k : \mathbb{R}^n \to \mathbb{R}^{d_{out}\times d_{in}}$ of a G-steerable convolution that maps between spaces of feature fields with representations ρ_{in} and ρ_{out}. The kernel itself is a function whose input in $\mathbb{R}^n$ transforms under the *standard representation* ρ_{st} (as defined in Section 2.1) and which we will model with an MLP. Since MLPs typically output vectors, it is convenient to *vectorize* the $d_{out} \times d_{in}$ output of the kernel. We denote the column-wise vectorization of a matrix $M \in \mathbb{R}^{d_1\times d_2}$ as $vec(M) \in \mathbb{R}^{d_1 d_2}$. Hence, we re-write the kernel in the following vector form:

$$vec(k(\cdot)) : \mathbb{R}^n \to \mathbb{R}^{d_{out} d_{in}} \tag{5}$$

Let $\otimes$ denote the *Kronecker product* between two matrices. Then, $\rho_{\otimes}(g) := \rho_{in}(g) \otimes \rho_{out}(g)$ is also a representation[4] of G. We suggest an implicit representation of the vectorized kernel k using an G-equivariant MLP $\phi : \mathbb{R}^n \to \mathbb{R}^{d_{out} d_{in}}$ based on the following lemma:

Lemma 1. *If a kernel k is parameterized by a G-equivariant MLP ϕ with input representation ρ_{st} and output representation $\rho_{\otimes} := \rho_{in} \otimes \rho_{out}$, i.e. $vec(k)(x) := \phi(x)$, then the kernel satisfies the equivariance constraint in Equation 4 for a compact group G.*

Proof. By construction, the equivariant MLP satisfies

$$\phi(\rho_{st}(g)x) = (\rho_{in}(g) \otimes \rho_{out}(g))\, \phi(x) \quad \forall g \in G, x \in \mathbb{R}^n \tag{6}$$

We further use the substitution and $\phi \mapsto vec(k(\cdot))$ and find:

$$vec(k(g.x)) = (\rho_{in}(g) \otimes \rho_{out}(g)) vec(k(x)) \tag{7}$$

Now, we make use of the following identity describing the vectorization of a product of multiple matrices:

$$vec(ABC) = (C^T \otimes A)\, vec(B) \tag{8}$$

[2] See Appendix A for the definition of a semi-direct product.

[3] To remain in the scope of the manuscript, we abstain from the mathematical definition of compact groups, which requires introducing topological groups. One can find more information about compact groups in [20].

[4] This representation is formally known as the *tensor product* of the two representations.

Hence, identity 8 allows us to re-write the previous equation as follows:

$$k(g.x) = \rho_{out}(g)k(x)\rho_{in}(g)^T \tag{9}$$

Since we assume G to be compact, its representations can always be transformed to an orthogonal form for which $\rho(g)^T = \rho(g)^{-1}$ via a change of basis. Hence, we find the equivariance constraint defined in equation 4. □

3.2. Comparison with [7]

[7, Theorem 2.1] describes a basis for any G-steerable kernel. It is useful to note that it essentially relies on two ingredients: *i)* a (pre-defined) finite G-steerable basis [14] for *scalar* filters and *ii)* a learnable equivariant linear map. A finite G-steerable basis is essentially a collection of B orthogonal functions, i.e. $Y : \mathbb{R}^n \to \mathbb{R}^B$, with the following equivariant property: $Y(g.x) = \rho_Y(g)Y(x)$, for some representation ρ_Y of G. The linear map in *ii)*, then, is a general equivariant linear layer, whose input and output transform as ρ_Y and $\rho_{in} \otimes \rho_{out}$. In comparison, the solution proposed in Section 3.1 replaces the pre-defined basis Y with a lernable MLP. More precisely, one can interpret its last linear layer as the map in *ii)*, and its activations before this layer as a learnable version of the basis Y in *i)*.

Since the design of a G-steerable basis is not straightforward, [7] suggests reusing a $O(n)$ steerable basis for any subgroup $G \subset O(n)$, which can be suboptimal. A G-steerable basis, by definition, must span a space of filters closed under G, i.e. for each filter $f : \mathbb{R}^n \to \mathbb{R}$, it should also contain $g.f$ for each $g \in G$. Because, in practice, the basis is finite dimensional, picking an $O(n)$-steerable basis might waste basis elements to be able to represent all rotated or mirrored functions, even if this is not necessary for $G \subset O(n)$ steerability. For example, if $n = 3$, an $O(3)$-steerable basis has a local support inside a sphere, which is suitable for the group of 3D rotations $SO(3)$ but only suboptimal for cylindrical symmetries, i.e. when G is the group $SO(2)$ of planar rotations around Z axis.

3.3. Expanding the input

Note that in the case of standard steerable convolutions, the input $x \in \mathbb{R}^n$ of a kernel is usually only the difference between the spatial position of two points. However, there is no requirement that would disallow the expansion of the input space except for practical reasons. Hence, here we augment steerable kernels with an additional feature vector $z \in \mathbb{R}^{d_z}$. This formulation allows us to incorporate relevant information in convolutional layers, such as physical and geometric features. For example, when performing convolutions on molecular graphs, z can encode the input and output atoms' types and yield different network outputs for different atoms. If introducing additional arguments into a kernel, the steerability constraint in equation 4 should be adapted to account for transformations of z:

$$k(g.x, \rho_z(g)z) = \rho_{out}(g)k(x, z)\rho_{in}(g)^{-1} \tag{10}$$

which must hold for all $g \in G, x \in \mathbb{R}^n, z \in \mathbb{R}^{d_z}$, where $\rho_z : G \to \mathbb{R}^{d_z \times d_z}$ is the representation of G acting on the additional features.

Again, analytically solving the constraint 10 to find a kernel basis for arbitrary ρ_z is generally unfeasible. Note also that the solution strategy proposed in [7] requires a basis for functions over $\mathbb{R}^{n+d_z}$, whose size tends to grow exponentially with d_z and, therefore, is not suitable. Alternatively, we can now use the flexibility of neural representation and introduce additional features into a kernel at no cost.

3.4. Implementing a G-equivariant MLP

We now describe how an MLP that is equivariant to the transformations of group G can be built. Equivariant MLPs have been generally described in previous works such as [13, 28, 35] and consist of a sequence of equivariant linear layers, alternated with non-linearities.

[13] describes a general algorithm to parameterize G-equivariant linear layers, provided the generators of the group and its Lie algebra are known. Alternatively, since the irreducible representations of G are typically known,[5] one can always rely on the following properties: *1)* any representation of a compact group can be decomposed as a direct sum of irreducible representations with a change of basis ([7] describes and implements a numerical method for this) and *2)* Schur's Lemma, which states that there exist equivariant linear maps only between irreducible representation of the same kind[6]. Hence, one can apply the right change of basis to the input and output of a linear layer and, then, learn only maps between input and output channels associated with the same irreducible representations. In particular, the tensor product representation $\rho_{in} \otimes \rho_{out}$ in the last layer is decomposed by a matrix containing the Clebsch-Gordan coefficients, which often appear in the analytical solutions of the kernel constraint in the related works. Note that the non-linearities used in the Steerable CNNs are G-equivariant and, therefore, can be used for the MLP too.

3.5. G-steerable point convolutions

Point convolutions operate on point clouds - sets of points endowed with spatial information:

$$X = (x_0, x_1, ..., x_{n-1}) \in \mathbb{R}^{3 \times n} \tag{11}$$

[5] This is a reasonable assumption since irreducible representations are often used to construct other representations used in Steerable CNNs. If this is not the case, there exist numerical methods to discover them from the group algebra.

[6] Typically, these maps only include scalar multiples of the identity matrix.

A point cloud provides a natural discretization of the data domain, which renders a convolution operator as follows:

$$(k * f_{in})(x_i) = \sum_{0 \leq j \leq n-1} k(x_i - x_j) f_{in}(x_j) \qquad (12)$$

In this formulation, the computational cost of the convolution scales quadratically with the number of nodes which might render the model infeasible to compute for large objects. A common strategy is to induce connectivity onto $x \in X$ and represent a point cloud X as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which is a collection of nodes $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ between pairs of nodes. Each node $v_i \in \mathcal{V}$ then is attached with spatial location x_i and a corresponding feature map $f(x_i)$ (see Figure 1). Each edge $e_{ij} \in \mathcal{E}$ can also be assigned with an attribute vector z_{ij} as in Section 3.3. A point convolution thus takes the form of learnable message-passing:

$$(k * f_{in})(x_i) = \sum_{j \in \mathcal{N}(i)} k(x_i - x_j, z_{ij}) f_{in}(x_j) \qquad (13)$$

where $\mathcal{N}(i) = \{j : (v_i, v_j) \in \mathcal{E} \text{ or } i = j\}$. Here, the computational cost scales linearly with the number of edges.

4. Related works

Group convolutions. Multiple architectures were proposed to achieve equivariance to a certain symmetry group. It has been proven [20] that the convolutional structure is a sufficient condition for building a model that is equivariant to translations and actions of a compact group. One can separate group convolutions into two classes depending on the space on which the convolution operates: *regular* [3, 4, 9, 12, 20] and *steerable* [7, 10, 11, 37, 39, 41, 43] group convolutions. In the first case, the input signal is represented in terms of scalar fields on a group G, and the convolution relies on a discretization of the group space. Steerable convolutions are a class of G-equivariant convolutions that operate on feature fields over homogeneous spaces and achieve equivariance via constraining the kernel space. They further avoid the discretization of the group space and can reduce the equivariance error in the case of continuous groups.

G-steerable kernels. In the case of rigid body motions $G = SO(3)$, the solution of the equivariance constraint is given by spherical harmonics modulated by an arbitrary continuous radial function, which was analytically obtained by Weiler *et al*. [41]. Lang and Weiler [21] then applied the Wigner-Eckart theorem to parametrize G-steerable kernel spaces over orbits of a compact G. The approach was later generalized by Cesa *et al*. [7], who proposed a solution for any compact sub-group of $O(3)$ based on group restriction. Using this approach, one can obtain a kernel basis for a group $G \leq H$ if the basis for H is known. Despite its generalizability, the method still requires a pre-defined basis for the group H that is further adapted to G. The resulting solution is not guaranteed to be optimal for G, see Section 3.2. We note that the practical value of steerable kernels is also high as they can be used for convolution over arbitrary manifolds in the framework of Gauge CNNs [8, 18, 40].

Implicit kernels. Using the implicit representation of convolution kernels for regular CNNs is not novel. It was used, for example, to model long-range dependencies in sequential data [30] or for signal representation [36]. Romero *et al*. [30] demonstrated that such parametrization allows building shallower networks, thus requiring less computational resources to capture global information about the system. In the area of equivariant convolutions, Finzi *et al*. [12] proposed parametrizing convolutions on Lie groups as continuous scalar functions in the group space. The method relies on a discretization of a continuous group, which might lead to undesirable stochasticity of the model's output. Instead, we use Steerable CNNs that define the kernel as a function on a homogeneous space. While the discretization of this space is still required, in most cases, it is naturally given by the data itself, e.g. in the case of point clouds; hence, no sampling error arises.

Equivariant point convolutions. A particular momentum has been gained by point convolutions in the form of equivariant message-passing [5, 31, 32, 37] specifically for problems where symmetry provides a strong inductive bias such as molecular modelling [1] or physical simulations [15]. Thomas *et al*. [37] pioneered $SE(3)$-equivariant steerable convolutions whose kernels are based on spherical harmonics modulated by a radial function. The approach was further generalized by Batzner *et al*. [2], who uses an MLP conditioned on the relative location to parameterize the radial function, although the basis of spherical harmonics is preserved. Brandstetter *et al*. [5] demonstrated that introducing geometric and physical information into an equivariant message-passing model improves the expressivity on various tasks. Following the direction, we demonstrate how implicit kernel-based steerable convolutions allow one to incorporate additional information into a model and significantly improve its performance. Note that, for $n = 3$ and $G = SO(3)$ or $O(3)$ and without additional edge features z, our MLP can only learn a function of the radius and, therefore, is equivalent to the models proposed in [2].

5. Experiments

In this section, we implement Steerable CNNs with implicit kernels (section 5.1) and apply it to point cloud data (ModelNet-40) (section 5.2) and molecular graph data

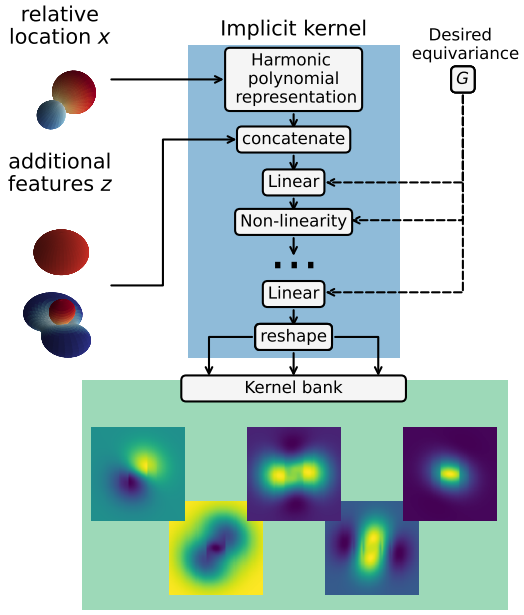

Figure 2. A schematic overview of the proposed implicit representation of steerable convolutional kernels. A G-equivariant MLP (blue) receives steerable vectors (here, the relative location x and additional features z) as input and returns a kernel bank (green) satisfying the equivariance constraint imposed by the group G.

(QM9) (section 5.3). First, we demonstrate how the framework can be easily applied to different symmetry groups G. The experiment proves the generalizability of the proposed approach as well as the gain in performance compared to the method proposed in [7]. Afterwards, we show that one can introduce additional physical information into a kernel and significantly improve the performance of steerable convolutions on molecular data. At last, we indicate the overall performance on regression tasks on the QM9 dataset.

5.1. Implementation

Implicit kernels. The schematic overview of an implicit kernel parameterized by a G-equivariant MLP (see section 3.4) is shown in Figure 2. The MLP takes as input a steerable vector obtained via direct sum (concatenation) of harmonic polynomial representation of the relative location x and (for QM9 experiments, section 5.3) additional features. We also batch-normalize the resulting feature field before passing it to the first linear layer; see details on preprocessing in Appendix B.1. Except for the last one, each layer is followed by a quotient ELU non-linearity [7]. The last layer generates a steerable vector which we reshape to yield a final convolutional kernel. Before training a model, we evaluate the implicit kernel of each layer at multiple points and compute the standard deviation of its output. We further divide the output of an MLP by this standard deviation during a forward pass which can be seen as a form of generalized He initialization [42]. We used the `escnn` library [7] to implement all our models.

Steerable convolutions. We implement each model as a stack of steerable convolutional layers followed by spherical quotient ELU non-linearities. The last steerable layer returns a vector with invariant features for each node, to which we apply global pooling to obtain a global representation of an object. We compare the implicit kernel-based architecture (equation 13) with the one that uses kernel basis (equation 12) developed in [7]. For each task, we tune the hyperparameters of a model with implicit kernels on validation data: the number of layers, the number of channels in each layer, the depth and the width of implicit kernels, and the number of training epochs. For a fair comparison of the aforementioned methods, we use the same number of layers but vary the number of channels in layers. This way, we yield a similar number of parameters of each model with a deviation of less than 10%. For ModelNet-40 experiments, our architecture is partially inspired by the model introduced in [27], which uses a gradual downsampling of a point cloud[7]. For QM9 experiments, we add residual connections to each steerable layer to learn higher-frequency features [17]. Details about optimization and model implementation for each task can be found in Appendix B.2.

5.2. Point cloud data: ModelNet-40

Dataset. The ModelNet-40 [45] dataset contains 12311 CAD models from the 40 categories of furniture with the orientation of each object manually aligned. 2468 models are reserved for the test partition. From the remaining objects, we take 80% for training and 20% for validation. We augment each partition, including the test one, with random rotations around the Z-axis. We induce connectivity on the point cloud by applying a k-nearest neighbour search with $k = 15$ at each model layer.

Depth-width trade-off. Romero *et al*. [30] indicated that implicit representation of convolutional kernels allows one to build a shallower model compared to standard CNNs. We, however, did not obtain a similar pattern. Even though the width of standard steerable convolutional kernels must be pre-specified, keeping it sufficiently large yields the same scaling pattern as for implicit ones. We note that implicit kernels can adapt their width and thus the field of view, which is not the case for standard steerable kernels. We hypothesize that the result might change on different datasets where long-range dependencies play a more important role, e.g. in sequential data as shown in [30].

[7] Poulenard *et al*. [27] use kd-tree pooling to compute coarser point clouds, while we use random sampling of points. Note also that the spherical quotient ELU we used is similar in spirit to the functional non-linearity described there.

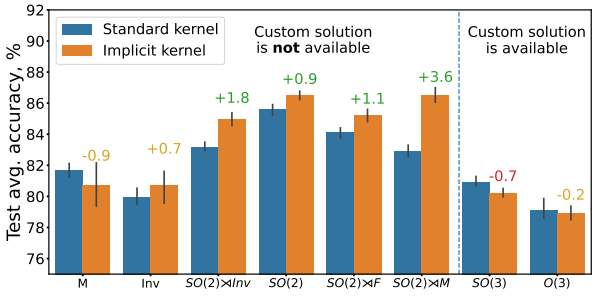

Figure 3. Performance comparison of Steerable CNNs with different kernels on the rotated ModelNet-40 dataset for different G. Bars denote mean average accuracy on the test dataset with error bars corresponding to standard deviation; both computed on 5 runs. The numbers above bars denote the performance gain of implicit kernels (orange) with respect to the ones obtained in [7] (blue). We depict a statistically significant (p-value < 0.05) positive difference with green and negative with red. Statistically insignificant (p-value > 0.05) difference is depicted with yellow. The group $SO(2)$ includes planar rotations around the Z axis, M and Inv contain respectively mirroring along the X axis and the origin. F contains rotations by π around the X axis. The group $O(2) \cong SO(2) \rtimes M$ represents the real symmetries of the data and, indeed, achieves the best accuracy.

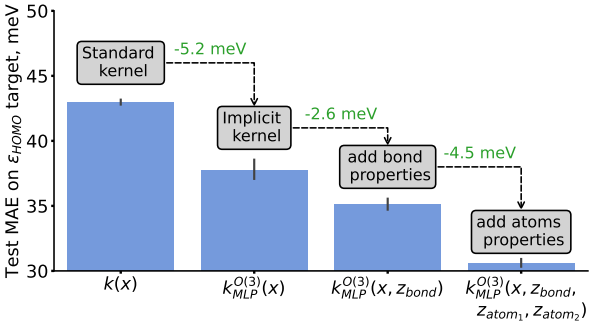

Figure 4. Using implicit kernels k^G_{MLP} and injecting it with bond and atoms properties significantly improves the performance of Steerable CNNs on the QM9 dataset (Mean Absolute Error on the ε_{HOMO} regression problem). Bars denote mean average accuracy on the test dataset with error bars corresponding to standard deviation; both computed on 5 runs. The kernel satisfies the constraint of the $O(3)$ group rendering the final architecture $E(3)$-equivariant.

Generalizability of implicit kernels. In this section, we demonstrate how one can build a steerable CNN that is equivariant to an arbitrary subgroup of the Euclidean group $G \rtimes (\mathbb{R}^3, +) \leq E(3)$. We compare the performance of implicit kernels with the standard steerable kernels obtained by group restriction [7] and keep the number of parameters similar. The results are shown in Figure 3. Implicit kernels achieve significant improvement in accuracy on test data for the majority of groups. The only statistically significant negative difference is presented for $G = SO(3)$, for which a tailored and hence optimal kernel basis is already available. When a custom solution is unknown, implicit kernels often significantly outperform the previously proposed method. Hence, they pose an efficient toolkit for building a kernel basis for an arbitrary subgroup of $E(3)$.

5.3. Molecular data: QM9

Dataset. The QM9 dataset [44] is a public dataset consisting of about 130k molecules with up to 29 atoms per molecule. Each molecule is represented by a graph with nodes denoting atoms and edges denoting covalent bonds. Each node is assigned a feature vector consisting of one-hot encoding of the type of the respective atom (H, C, N, O, F) and its spatial information corresponding to a low energy conformation. Additionally, each molecule is described by 19 properties from which we select 12 commonly taken in literature [5] for regression tasks. Different from common practice, we perform convolution on molecular graphs, i.e. with connectivity pre-defined by molecular structure instead of inducing it. The design choice is motivated by the presence of edge features which we include in an implicit kernel to display the flexibility of neural representation.

Flexibility of implicit kernels. We first demonstrate how one can use the flexibility of neural representation to introduce additional features of choice into a convolutional layer. As each pair of atoms connected by a covalent bond is assigned a one-hot encoding of the bond type z_{ij}, we use it as a condition for the $E(3)$-equivariant MLP. Additionally, we follow Musaelian *et al.* [26] and embed one-hot encoding of the center and neighbor atom types z_i and z_j into the MLP. We include each property one by one and indicate corresponding performance gain in Figure 4. First, we observed a sharp improvement by switching from standard steerable kernels to implicit ones. We attribute it to the higher expressivity of implicit kernels and their ability to learn more complex interactions. Furthermore, injecting scalar attributes into the kernel computation reduced MAE even further. Introducing both atom types and an edge type significantly influenced the performance, which corresponds to the model learning how to process each specific combination differently, thus adding to its expressivity. It does not come as a surprise, as a similar result was obtained by Brandstetter *et al.* [5], who used non-linear message aggregation conditioned on physical information and demonstrated the corresponding yield in performance.

Performance on molecular data. Table 1 shows the results of an $E(3)$-equivariant steerable CNN with implicit kernels on the QM9 dataset. For non-energy regression tasks (α, $\Delta\varepsilon$, ε_{HOMO}, ε_{LUMO}, μ and C_ν), we obtain results that are comparable with the ones of the best per-

Table 1. Mean Absolute Error (MAE) between model predictions and ground truth for the molecular property prediction on the QM9 dataset. Linear steerable convolutions are denoted by $*$.

Task	α	$\Delta\varepsilon$	ε_{HOMO}	ε_{LUMO}	μ	C_ν	G	H	R^2	U	U_0	ZPVE
Units	bohr^3	meV	meV	meV	D	cal/mol K	meV	meV	bohr^3	meV	meV	meV
NMP [16]	.092	69	43	38	.030	.040	19	17	0.180	20	20	1.50
SchNet [32]	.235	63	41	34	.033	.033	14	14	0.073	19	14	1.70
Cormorant [1]*	.085	61	34	38	.038	.026	20	21	0.961	21	22	2.02
L1Net [25]*	.088	68	46	35	.043	.031	14	14	0.354	14	13	1.56
LieConv [12]	.084	49	30	25	.032	.038	22	24	0.800	19	19	2.28
TFN [37]*	.223	58	40	38	.064	.101	-	-	-	-	-	-
SE(3)-Tr. [15]	.142	53	35	33	.051	.054	-	-	-	-	-	-
DimeNet++ [19]	.043	32	24	19	.029	.023	7	6	0.331	6	6	1.21
SphereNet [23]	.046	32	23	18	.026	.021	8	6	0.292	7	6	1.12
PaiNN [33]	.045	45	27	20	.012	.024	7	6	0.066	5	5	1.28
EGNN [31]	.071	48	29	25	.029	.031	12	12	0.106	12	12	1.55
SEGNN [5]	.060	42	24	21	.023	.031	15	16	0.660	13	15	1.62
Ours	.091	48	26	26	.034	.037	30	39	1.493	35	32	2.34

forming approaches. We also indicate that for these variables, a steerable CNN with implicit kernels outperforms steerable linear convolutions (TFN, LieConv, L1Net) on most tasks. However, this does not generalize to the remaining energy variables (G, H, U, U_0, ZPVE) and R^2, where the model significantly falls behind all the benchmark approaches. We hypothesize that it can be attributed to two factors. First, steerable convolutions generally do not perform well on these tasks compared to problem-tailored frameworks (PaiNN, DimeNet++, SphereNet), even in the non-linear case (SEGNN). Second, we hypothesize that molecular connectivity does not produce a sufficient number of atom-atom interactions. This is supported by the evidence obtained by Brandstetter *et al*. [5], which indicates that the total number of messages drastically improves the model performance. However, as the goal of the section was to demonstrate the flexibility of implicit kernels that can be conditioned on features of graph edges, we leave developing more involved architectures (e.g. with induced connectivity) for further work.

6. Conclusion

In this work, we propose a novel approach for implementing convolutional kernels of Steerable CNNs equivariant to translations and actions of an arbitrary compact group G. We utilize a G-equivariant MLP to parameterize a G-steerable kernel basis. We theoretically prove that the equivariance of an MLP is a sufficient condition for building an equivariant steerable convolutional layer. We validate the generalizability of the proposed framework on point cloud data; we demonstrate that the implicit representation allows one to implement equivariance to various groups for which a custom kernel basis has not been developed. We also show that implicit kernels significantly outperform a previously proposed general method. Moreover, the flexibility of neural representation allows one to introduce additional arbitrary features into a convolutional kernel to improve the overall expressivity of a steerable CNN. We indicate that the property is ultimately beneficial by applying Steerable CNNs to molecular data. Overall, we suggest a simple yet efficient solution for building a general kernel basis for an arbitrary symmetry group.

References

[1] Brandon M. Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *NeurIPS*, 2019. 5, 8

[2] Simon Batzner, Tess E. Smidt, Lixin Sun, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13, 2022. 5

[3] Erik J. Bekkers. B-spline cnns on lie groups. *ArXiv*, abs/1909.12057, 2020. 2, 5

[4] Erik J. Bekkers, Maxime W Lafarge, Mitko Veta, Koen A.J. Eppenhof, Josien P.W. Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2018. 5

[5] Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and physical quantities improve e(3) equivariant message passing. *ArXiv*, abs/2110.02905, 2022. 2, 3, 5, 7, 8

[6] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velivckovi'c. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478, 2021. 1

[7] Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build e(n)-equivariant steerable cnns. In *ICLR*, 2022. 2, 3, 4, 5, 6, 7

[8] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2019. 5

[9] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016. 1, 2, 5

[10] Taco S. Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. *Advances in neural information processing systems*, 32, 2019. 2, 5

[11] Taco S. Cohen and Max Welling. Steerable CNNs. In *ICLR 2017*, Nov. 2016. 2, 5

[12] 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. 2, 5, 8

[13] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *ArXiv*, abs/2104.09459, 2021. 4

[14] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991. 4

[15] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. *ArXiv*, abs/2006.10503, 2020. 5, 8

[16] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, abs/1704.01212, 2017. 8

[17] Francesco Di Giovanni, James R. Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael Bronstein. Graph neural networks as gradient flows: understanding graph convolutions via energy. 2022. 6

[18] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh {cnn}s: Anisotropic convolutions on geometric graphs. In *International Conference on Learning Representations*, 2021. 5

[19] Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Gunnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *ArXiv*, abs/2011.14115, 2020. 8

[20] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *ArXiv*, abs/1802.03690, 2018. 2, 3, 5

[21] Leon Lang and Maurice Weiler. A Wigner-Eckart theorem for group equivariant convolution kernels. In *International Conference on Learning Representations*, 2020. 5

[22] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. 1998. 1

[23] Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d graph networks. *ArXiv*, abs/2102.05013, 2021. 8

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 11

[25] Benjamin Kurt Miller, Mario Geiger, Tess E. Smidt, and Frank No'e. Relevance of rotationally equivariant convolutions for predicting molecular properties. *ArXiv*, abs/2008.08461, 2020. 8

[26] 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. *ArXiv*, abs/2204.05249, 2022. 7

[27] Adrien Poulenard and Leonidas J. Guibas. A functional approach to rotation equivariant non-linearities for tensor field networks. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13169–13178, 2021. 6

[28] Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. *arXiv preprint arXiv:2002.02912*, 2020. 4

[29] David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021. 2

[30] David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *ArXiv*, abs/2102.02611, 2022. 2, 5, 6

[31] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *ICML*, 2021. 3, 5, 8

[32] 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 *NIPS*, 2017. 5, 8

[33] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*, 2021. 8

[34] Jean-Pierre Serre. Linear representations of finite groups. 1977. 2

[35] J. Shawe-Taylor. Building symmetries into feedforward networks. In *1989 First IEE International Conference on Artificial Neural Networks, (Conf. Publ. No. 313)*, page 158–162, Oct 1989. 4

[36] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *ArXiv*, abs/2006.09661, 2020. 5

[37] Nathaniel Thomas, Tess E. Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick F. Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *ArXiv*, abs/1802.08219, 2018. 5, 8

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019. 11

[39] Maurice Weiler and Gabriele Cesa. General $E(2)$-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 2, 3, 5

[40] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. Coordinate Independent Convolutional Networks - Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds. *arXiv preprint arXiv:2106.06020*, 2021. 3, 5

[41] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018. 2, 3, 5

[42] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[43] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[44] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *arXiv: Learning*, 2017. 2, 7

[45] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 2, 6

A. Additional details and definitions on group theory

Definition 5 (Semi-direct product)**.** *Let N and H be two groups each with their own group product which we denote with the same symbol $\cdot$, and let H act on N by the action $\odot$. Then a (outer) semi-direct product $G = N \rtimes H$, called the semi-direct product of H acting on N , is a group whose set of elements is the Cartesian product $N \times H$, and which has group product and inverse:*

$$(n, h) \cdot (\hat{n}, \hat{h}) = (n \cdot (h \odot \hat{n}), h \cdot \hat{h})$$

$$(n, h)^{-1} = (h^{-1} \odot n^{-1}, h^{-1})$$

for all $n, \hat{n} \in N$ and $h, \hat{h} \in H$.

B. Experimental details

B.1. Preprocessing kernel's input

An implicit kernel receives as input the relative location x and additional features z. The first argument is a set of 3-dimensional points transforming according to the standard representation ρ_{st}. We first batch-normalize them and then scale to range $[0, 1]$ by using the squashing function $x/(1 + x)$. Afterwards, we compute its *homogeneous polynomial representation* in $\mathbb{R}^3$ up to order L. The harmonic polynomial $Y_l(x)$ of order l evalutated on a point $x \in \mathbb{R}^3$ is a $2l + 1$ dimensional vector transforming according to the Wigner-D matrix of frequency l (and parity $l \mod 2$, when interpreted as an irrep of $O(3)$). The vector $Y_l(x)$ is computed by projecting $x^{\otimes l} \in \mathbb{R}^{3^l}$ on its only $2l+1$ dimensional subspace transforming under the frequency l Wigner-D matrix.[8] We keep $L = 3$ as we found the choice to be favourable for overall performance on validation data. The second argument is a vector of scalar values corresponding to trivial representations, which we concatenate to the harmonic representation. We extensively compared different preprocessing techniques, and the combination of batch normalization and squashing of relative location improved the performance of implicit kernels the most. We attribute it to higher numerical stability as we discovered that the standard deviation of MLP's output is the lowest in the case.

The output of the implicit representation is a vector which we multiply with a Gaussian radial shell $\phi(x) = exp(-0.5 \cdot ||x||_2^2/\sigma^2)$ where σ is a learnable parameter.

[8] If D_l is the frequency l Wigner-D matrix, x transforms under D_l, which is isomorphic to the standard representation of $SO(3)$. Then, $x^{\otimes l}$ transforms under $D_1^{\otimes l}$, i.e. the tensor product of l copies of D_1. The tensor product of two Wigner-D matrices is well known to decompose as $D_l \otimes D_j \cong \bigoplus_{i=|l-j|}^{l+j} D_i$. By applying this rule recursively, one can show that $D_1^{\otimes l}$ contains precisely one copy of D_l. We define $Y_l(x)$ as the linear projection of $x^{\otimes l}$ to this subspace. Note also that, since this is a linear projection, the definition of Y_l satisfies the defining property of *homogeneous polynomials* $Y_l(\lambda x) = \lambda^l Y_l(x)$.

Table A1. Number of channels in each convolutional layer of a steerable CNN (see section 5.2).

G	kernel	channels	#par, $\cdot 10^3$
M	Implicit	20 20 20 20 20 128	122
	Standard	20 20 20 20 20 128	144
Inv	Implicit	20 20 20 20 20 128	122
	Standard	20 20 20 20 20 128	144
$SO(2) \rtimes Inv$	Implicit	11 11 12 12 12 128	588
	Standard	20 25 25 30 30 128	557
$SO(2)$	Implicit	15 15 20 30 30 128	565
	Standard	30 30 40 40 40 128	561
$SO(2) \rtimes F$	Implicit	12 12 12 12 12 128	580
	Standard	20 25 25 30 30 128	557
$SO(2) \rtimes M$	Implicit	15 20 20 20 20 128	592
	Standard	30 40 40 40 40 128	592
$SO(3)$	Implicit	10 10 10 10 20 128	160
	Standard	30 30 30 30 30 128	154
$O(3)$	Implicit	10 10 10 10 20 128	140
	Standard	30 30 30 30 30 128	128

We note that in our case, all additional attributes z (QM9 experiment) were one-hot encodings. They, thus, did not demand further transformation. We assume, however, that normalizing is preferable in the general case.

B.2. Model implementation

ModelNet-40 We trained each model using batch size 32 for 200 epochs which we found to be sufficient for convergence. We minimized the cross entropy loss with label smoothing [38]. We used AdamW optimizer [24] with an initial learning rate 10^{-3}. We also decayed the learning rate by 0.5 after 25, 50 and 75 epochs. The run-time on average was 3h on an NVIDIA Tesla V100 GPU and varied across different G .

For the final experiments described in section 5.2, we keep the configuration of G-equivariant MLP the same - 2 linear layers with 8 copies of the same representation and spherical quotient ELU with maximum frequency of 2 in between. Each model consists of 6 steerable convolutions followed by spherical quotient ELU and batch normalization. The first layer takes coordinates of each point in point cloud with the standard representation as input. We employ steerable vector spaces up to order 2 in each convolutional layer. We only varied the number of channels in each layer to achieve a similar overall number of parameters for the models compared. We indicate the number of channels for each layer and the total number of parameters for each model in Table A1. The last layer produces a vector of 128 scalar features for each node. We apply global max pooling yielding a 128-dimensional embedding of a point cloud. We further use 2-layer MLP ($128 \xrightarrow{ELU} 128 \xrightarrow{ELU} 40$) to compute the probability of a point cloud to belong to a class.

QM9 Each model has the following structure: embedding layer $\rightarrow$ steerable convolutional layers $\rightarrow$ global mean pooling $\rightarrow$ 2-layer MLP. The embedding layer consists of 3 parts: linear layer $\rightarrow$ learnable tensor product $\rightarrow$ spherical quotient ELU. First, it takes one-hot encoding of an atom type and applies a linear transformation. The learnable tensor product computes the tensor product of each field with itself to generate an intermediate feature map. Then, a learnable linear projection is applied to the feature map, which yields a map from trivial representations to spherical representations up to order $L = 2$. We use steerable feature vectors with the maximum order of 2 in each convolutional layer. The final classification MLP is defined as follows: $128 \xrightarrow{ELU} 128 \xrightarrow{ELU} 1$.

We optimized the number of layers and channels for the ε_{HOMO} regression task. We trained each model using batch size 128 for 130 epochs. Each model is optimized with AdamW optimizer with an initial learning rate of $5 \cdot 10^{-4}$. We use learning rate decay by 0.5 after 25, 50, 75, 100 and 125 epochs. It took 40h to train a model on an NVIDIA Tesla V100 GPU. We employ steerable CNNs with 11 convolutional layers with residual connections and 24 channels for the flexibility experiment. For the final performance indicated in Table 1, we scale the model up and use 15 layers. Implicit kernels are parameterized with $O(3)$-equivariant MLP with 2 linear layers with 16 channels and spherical quotient ELU in between.