Review of Rotational Equivariant Convolution Neural Networks

Wenhao Zhang
National Institute of Materials Science, Tsukuba

***Abstract*—Convolution neural network are very successful in that, through weight sharing convolution and pooling, higher and higher abstraction of the original input are can be generated approximately equivariantly, i. e., translation properties of the input (in most case image) are kept until at late stage. Weight sharing also greatly reduced the number of parameter, making it easier to be trained. Recently, there are many attempts to generalize the standard convolution network and make them rotational equivariant so that the information about how an image is rotated is also kept throughout the network until late stage. This is a highly desirable property for machine learning on materials structures. For example, if we want to predict the atomic force on the atom from its local environment. Such rotation equivariance because necessary because we want the predicted force to rotate in the same way as the input local environment (rotational information to flow through the network) under the same set of convolution filter. Apart from learning the force, other directional property, such as atomic orbitals that depend on orientation of local environments, can also be learnt (maybe indirectly as feature map). In this work, we provide a brief review of the concepts underlying rotational equivariant network and build up our understanding towards two successful network model: one operate on spherical image and another on set of points.**

I. Introduction

Neural networks (NN) are highly flexible regarding inputs and have been applied to many applications such as numerical data, images, sound and texts. NNs can usually achieve human level performance in tasks based on these kinds of data, such as image classification and translation. Therefore, it is very attractive if NNs apply to crystal structure inputs as well and achieve human level performance in chemistry. This is indeed the focus on many studies.

In this review, we describe the framework that operate on crystal structures. We start by describing networks that is invariance to transformations and describe the approaches that enable equivariance.

II. Invariant network on points and graphs

A crystal (molecule) structure is simply a set of points in space. To write it down, we usually define a basis and present all atoms in a list with their positions. Clearly, the structure is the same regardless of its orientation and in what sequence the atoms are arranged in the list. Different orientation of the input structure can be obtained by an affine transformation. Therefore, to describe properties based on input structure, we require the output to be invariance to permutation and affine transformations. In this section, we review two network structures that achieve these properties, with the focus on how invariance is achieved. One additional points that we need to consider in these networks is how the points interact with each other. In this kind of tasks, *no point is an island*, and we must let information flow from points to points. How this is achieved is also a focus in our following discussion.

A. Network on points

Qi et al.[1] proposed a network (PointNet) design that operation on sets of points that respect the permutation and transformation invariance of the input points. Although these networks are originally designed to classify objects from spatial scanning, they are general methods that treat with points in space. The main structure of their network are can be summarized by the following operation:

$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n)) \tag{1}$$

Here, function f is a *general* function that take a set of points and output some properties $f\colon$ set of points $\to \mathbb{R}$. In PointNet, this function is approximated using two functions. $h\colon \mathbb{R}^N \to \mathbb{R}^K$ is a multi-layer neural network and g is some function that is symmetric to permutation, such as averaging or max-pooling. Note that the same h is used for all points to ensure permutation invariance. In the first layer, x_i denote inputs of coordination (x, y, z), and they are expanded to feature vector in higher dimensional feature space by $h(x)$

To achieve invariance under affine transformation, they define a small module that take positions of all points as input and output an affine transformation matrix that are used to rotate all points. Therefore, this small module is expected to learn the 'standard' alignment of the points. This is the module used to achieve invariance under affine transformations, although the invariance is learned, not built-in.

The interaction between the points in PointNet (especially in the subsequent PointNet++[2]) is achieved as following: points are organized in a hierarchy (such as partitioned by their positions in space) and function equation (1) is used on the subset of points to extract some properties in the local area (for example, if these points belong to a flat surface). These properties can be passed to later networks in a hierarchical fashion, or feed back (concatenated) to the features on individual points to pass along to later networks. The affine transformation, being a global properties applied to the input points, is a specical case. Such a hierarchical structure is similar to convolution networks.

B. Messaging passing networks based on graph

A graph is represented as $G = (V, E)$ where V is a set of nodes and E is the set of edges[3]. Because the atoms in a molecule can be considered to bond to each other, it seems very natural that we represent a molecule structure using a graph, with atoms as nodes and bonds between atoms as edges. In this case, the absolute position of the atoms in space are no longer and their relative positional informations are converted to edge attributes. In most cases, the geometry of the structure is specified using the relative distance between atoms. The graph representation of a structure therefore remain the same under transformation of the input structure.

For crystals, a straightforward extension of molecular graph above is not applicable because of the periodicity. But we can define *crystal graph* by allowing multiple edges between two atoms in a unit cell. Each edge from A to B correspond to an actual edge from A to a periodic image B, within a cutoff radius.

In networks using graph, we store information h_i on nodes i and global properties are given typically by summing over all nodes. The graph structure and summation output ensures the invariance of output under permutation and affine transformation.

Message passing refers to the way how informations are passed from node i to node j through their edges with some attributes e_{ij}. Typically, message passing operates for T iterations. We call

$$m_i^{t+1} = \sum_{j \in neig(i)} M_t(h_i^t, h_j^t, e_{ij}) \tag{2}$$

a message[4], where t index the steps. M_t is called a *message function*. h_i^0 typically encode some atomic feature, such as atomic numbers. Node attributes h are main information that are updated during the iterations and get passed around:

$$h_i^{t+1} = U_t(h_i^t, m_i^{t+1}) \tag{3}$$

function U_t defines how the hidden states are updated depending on the information it received and is called *update function*.

Different purposed message passing networks use different message functions and update functions. For example, Gilmer et al. built MPNN[4] using

$$M_t(h_i^t, h_j^t, e_{ij}) = A(e_{ij})h_j \tag{4}$$

as message function where A maps an edge to a matrix to be multiplied with node feature h_j. They used Gated Recurrent Unit as the update function $U_t = \text{GRU}(h_i^t, m_i^{t+1})$ and a *set to set* operation introduced by Vinyals et al. 2015.

Schütt's deep tensor neural network (DTNN)[5] creates messages with so called 'interaction' between two atoms: v_{ij} as:

$$m_i^{t+1} = \sum_{j \neq i} v_{ij} \tag{5}$$

$$= \sum_{j \neq i} \tanh\left(W^{fc}[(W^{cf}h_j^t + b^{f_1}) \circ (W^{df}e_{ij} + b^{f_2})]\right) \tag{6}$$

where W and b are network adjustable parameters and $\circ$ indicate a element-wise product (Hadamard product). The node features are simply updated by $h_i^{t+1} = h_i^t + m_i^{t+1}$. The network ooutput total energies, which is a summation over 'atomic energies': $E = \sum_i E_i = \sum_i E(h_i^T)$ where E is a multi-layer perceptron that convert atomic feature into energies. Latter, Schütt et al. extended their work to crystals by using a filter that depend on the distance between atom i and all the periodic image of j[6].

Xie and Grossman purposed crystal graph convolution neural network**xie˙crystal˙2018** that use the previously mentioned crystal graph with multiple edges between two atoms due to periodicity. Denoting i, j as atom index and $(i, j)_k$ as the k^{th} bond between in the crystal graph, the message function are obtained as:

$$m_i^{t+1} = \sum_{j,k} \sigma(z_{(i,j)_k}^t W_f^t + b_f^t) \circ g(z_{(i,j)_k}^t W_s^t + b_s^t) \tag{7}$$

where $z_{(i,j)_k} = h_i \oplus h_j \oplus e_{(i,j)_k}$ and σ, g are non-linear functions. Hidden states are updated by addition $h_i^{t+1} = h_i^t + m_i^{t+1}$. A weighted sum is used as pooling operation to produce a weighted sum.

All the previous network use only distance between two atoms as edge feature, but directional feature are unutilized. The work by Klicpera et al.[7] (DimeNet) that attempt to use the directional information by using spherical basis functions. Their work is based on directed molecular graph and focus on the messages m_{ji} passing from atom j to i: $h_i^{t+1} = \sum_j m_{ji}^{t+1}$. The messages are updated by:

$$m_{ji}^{t+1} = F_{update}\left(m_{ji}^t, \sum_{k \in neig(j)} F_{int}(m_{kj}^t, e_{ji}^{RBF}, a_{kji}^{SBF})\right) \tag{8}$$

where atom k is an atom in the neighbor of j except i itself. e_{ji}^{RBF} is a embedding of distance and a_{kji}^{SBF} embedding the angle $\angle kji$ as well as distance $\|\mathbf{r_{kj}}\|$ using spherical functions. F_{update} and F_{int} are update functions and interaction functions, made of blocks of linear layers. DimeNet achieve one of the best performance on QM9 dataset, superior to SchNet.

III. Equivariant networks

Nueral networks based on graph achieve desired invariance as a built-in property of the graph input. But there are two disadvantage to the such networks: 1) invariance is desirable when we want the network to output some scalar global properties, but if we want to make prediction for directioinal properties (such as magnetization axis), the invariant network will not work. PointNet mentioned earlier is able to output rotations to align itself and can be used reversely to output directional properties. But this is not built-in to the network itself. PointNet is also not as communicative as message passing networks. 2) Many spatial informations are lost when we convert a structure into a graph, which can be used at later stage in the network. The directional message passing network keeps more information, but still not all.

This two problem can be overcame by an equivariant network. But before that, we review the notion of equivariance and how it is achieved from convolution networks.

A. Equivariance

Equivariance is a property of mapping which perserve the transformation properties from inputs to outputs. If we consider an operation in neural network as a function that map an input vector $f_{l-1} \in F_{l-1}$ to an output vector $f_l \in F_l$, in their corresponding vector space: $\Phi_l \colon F_{l-1} \to F_l$. Equivariance thus means that a transformation g in the input correspond to a transformation in the output vector space:

$$\Phi_l(T_g^{l-1} f_{l-1}) = T_g^l \Phi_l(f_{l-1}) \tag{9}$$

T_g are the linear transformation of g in corresponding vector space F_{l-1} and F_l such that $T_g = T_v T_u$ if $g = vu$ and they are called a representation of the group of transformation G.

Invariance is a special case of equivariance: if T_g^l is identity for all transformation g, we have the relationship:

$$\Phi_l(T_g^{l-1} f_{l-1}) = \Phi_l(f_{l-1}) \qquad \text{for all } g \tag{10}$$

i.e. All transformation on the input feature map leave the output feature map invariant.

As an example, consider an image with an arrow painted on it and we want to read the vector (x, y) that give the direction of the arrow. The image lives in the (high dimensional) vector space of all possible images and the arrow lives in on a plane. If we rotate the image by angle θ, we obtain another images. Of course, the vector we read out is also rotated by the same angle. Although trival, the neural network in our brain that convert the signal on our retina to a vector on a plane is equivariant. We introduce some important works below that purpose equivariant network structures. We start by the simplest case, the convolution network.

B. Structure of convolution network

Convolution neural networks (CNN) are one of the most successful examples of machine learning. They contain at least one layer with convolution operation. Although the network is named convolution, most of the implementation use correlation instead and we also use correlation in this article. A simple CNN contain stacked components of correlation, nonlinear activation and pooling. The intermediate signals are called *feature map* in CNNs. Fully connected linear layers are often placed before the output. What make CNNs stand out are[8] 1) sparse connection and parameter sharing make CNN efficient to store and train, 2) due to weight sharing, convolution operations are equivariant to translation and 3) pooling operation.

Correlation: We consider the case of images with a single channel(gray scale) and discrete pixels. First define *feature map* as a mapping $f \colon \mathcal{X} \to \mathbb{R}^n$ where $\mathcal{X}$ is a set of positions $x = (u, v)$ on the image indicating each pixel. A correlation is then $\Phi_l \colon F_{l-1} \to F_l$, where F_l is the vector space of feature maps at layer l. (F_0 is therefore the space of all possible input images). Φ_l is given by:

$$(\Phi_l \circ f_{l-1})(x) = (f_{l-1} \star g_l)(x) = \sum_{y=(u',v')} f_{l-1}(y) g_l(y - x) \tag{11}$$

where $\Phi_l \circ f_{l-1}$ means applying Φ_l to f_{l-1}, $g_l \colon \mathcal{X} \to \mathbb{R}^n$ is called the *filter*, ~~or *kernel*, of the correlation.~~

Pooling: A pooling layer extract a summary statistic of nearby outputs at a certain location. Different pooling can be used, such as average pooling and max-pooling. For example, max-pooling P_l^{max} can be defined as:

$$(P_l^{max} \circ f_{l-1})(x) = \max_{x' \in X, k \in K} f_{l-1}^k(x') \tag{12}$$

where X is set of points near x and K is a set of channel indexs. Pooling have two important functionality:

1) Pooling over x make the output feature map approximately invariant to small translations of the input (blurring effect).
2) Pooling over feature depth, we can achieve some degree of invariance to transformation, such as rotation.

They are illustrated by the following figure.1.

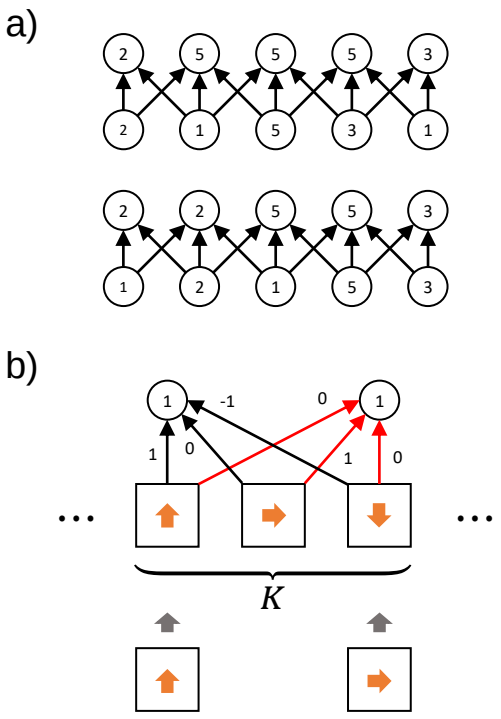

Fig. 1. a) All the inputs of the pooling is shifted right by one. However, only one output value is changed. This shows that the output of the pooling is less sensitive to translation. b) polling over the K channel with rotated filters give results invariance to the rotations

Equivariance of CNN on translation: The effect of a translation $t = (u_t, v_t)$ on the feature map f_{l-1} is given as:

$$(\Phi_t \circ f)(x) = f(x - t) \tag{13}$$

This is show simply in Figure.2

We can show that convolution (11) is translation equivariant,

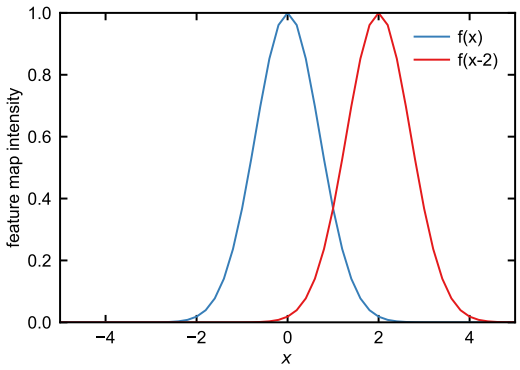

Fig. 2. Apply a shift to resulting feature map is equivalent to a reverse shift in x. In one-dimensional case with feature map given as $f : \mathbb{R} \to \mathbb{R}$, the feature map is translated to the right by $t = 2$, which correspond to the feature map $f(x - 2)$

satisfying Equation (9)

$$
\begin{aligned}
(\Phi_l \circ [\Phi_t \circ f_{l-1}])(x) &= \sum_y f_{l-1}(y-t) g_l(y-x) \\
&= \sum_{y'} f_{l-1}(y') g_l(y'-x+t) \\
&= \sum_{y'} f_{l-1}(y') g_l(y'-(x-t)) \\
&= (\Phi_l \circ f_{l-1})(x-t) \\
&= (\Phi_t \circ [\Phi_l \circ f_{l-1}])(x)
\end{aligned}
$$

Therefore, $(\Phi_l \circ [\Psi_t \circ f_{l-1}])(x) = (\Phi_t \circ [\Phi_l \circ f_{l-1}])(x)$ and the convolution is equivariant to translation. Similarly, both non-linear function and pooling are translation equivariant.

CNNs are not invariant: As shown in many studies, CNN predictions can be translation invariant: it can recognize the objects in image regardless of exact position of that object. However, we should distinguish the built-in equivariance from the learnt invariance[9]: by constructure, CNNs are equivariant to translation[1], but it does not necessary lead to invariance in the prediction. CNN can be trained efficiently to identifying objects at different locations on the image with good performance. The keypoint to achieve such invariance in the result seems to due to:

1) Effect of pooling with decreasing spatial resolution with higher depth of features (channels) that contain highly extracted information.
2) The larger field of reception at deeper layer, containing more spatial correlations.

C. Group equivariant convolution networks

Now, we generalize the standard CNN so that it is equivariant to general transformations. First, we will define transformation as elements in a group, then, we describe the group-equivariant version of CNN on discrete and continuous groups.

[1]as noted by E. K. Abrams, 2017, equivariance may be lost if pooling involve strides

Group of transformation: Transformations of an object form a group G since two successive transformations yield another transformation $g_1 g_2 \in G$ and we can always find a reverse transformation $g^{-1} g = e$, where e is identity. We can now formally define the action of g as linear transformation T_g^l on vector space of feature maps at layer l. The linear transformation and the feature map space form a representation of the transformation group G. Now, we can define an equivariant CNN on group G as follows: When the input on CNN is transformed as $f_0 \to T_g^0 f_0$ by $g \in G$, all features maps in the network: f_l transform as $f_l \to T_g^l f_l$.

We define the operation of g on feature maps as:

$$
(\Psi_g \circ f)(x) = f(g^{-1} x) \tag{14}
$$

This form is consistent with Equation (13). The standard CNN are equivariant to the group of translations, but not to rotations $r \in G$. This can be shown as follows:

$$
\begin{aligned}
(\Phi_l \circ \Phi_r \circ f_{l-1})(x) &= \sum_y f_{l-1}(y) g_l(ry - x) \\
(\Phi_r \circ \Phi_l \circ f_{l-1})(x) &= \sum_y f_{l-1}(y) g_l(y - r^{-1} x)
\end{aligned}
$$

This is shown in Figure 3

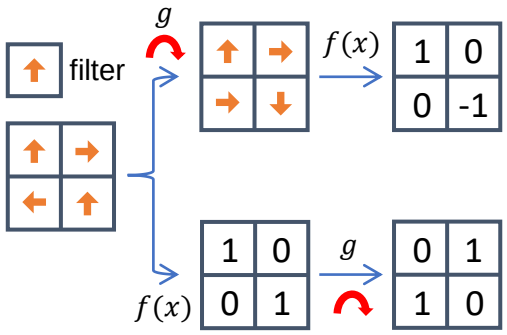

Fig. 3. Upper($[[\mathcal{L}_r f] \star \psi](x)$): we rotate the inputs, then perform correlation (rotating the feature map is equivalent to rotating the input with a reversed operation g^{-1}). Lower($[\mathcal{L}_t [f \star \psi]](x)$): we first perform correlation, then rotate the resulting feature map

Homogeneous space and Quotient group: We define a set $\mathcal{X}$ to be the homogeneous space of a group G if, taking any element $x_0 \in \mathcal{X}$, all other elements in $\mathcal{X}$ can be reached by applying $g \in G$ to x_0.

Choosing x_0, All elements leave x_0 invariant $\{g \mid g x_0 = x_0\}$ is called a stabilizer of x_0 and is a normal subgroup of G. For normal subgroup H of group G, gH is called a left coset for a $g \in G$. The set of all left cosets $\{gH \mid g \in G\}$ form a quotient group, denoted as G/H. The quotient group can be mapped to $\mathcal{X}$ by a bijection: the group elements in the same coset are mapped to the same $x \in \mathcal{X}$.

The key point here is to distinguish the group G from its homogeneous space. It is more natural to define the convolution on homogeneous space of the transformation, than on transformation itself. This will be shown in the following.

G-convolution for discrete transformation: We define convolution with respect to a group operation $u \in G$ (G-correlation) as:

$$(\Phi_l \circ f_{l-1})(u) = \sum_{v \in G} f_{l-1}(v) g_l(u^{-1}v) \tag{15}$$

where $h \in G$. We can see that the standard convolution Eq.(11) is a special case of the G-correlation.

Now, the feature map is defined on discrete trasnformation $f_l \colon G \to \mathbb{R}^{N_l}$, instead of $f \colon X \to \mathbb{R}^{N_l}$. Equation (15) is equivariant to transformation $g \in G$:

$$\begin{aligned}(\Phi_l \circ \Phi_g \circ f_{l-1})(u) &= \sum_{v \in G} f_{l-1}(g^{-1}v) g_l(u^{-1}v) \\ &= \sum_{v' \in G} f_{l-1}(v') g_l(u^{-1}gv') \\ &= \sum_{v' \in G} f_{l-1}(v') g_l((g^{-1}u)^{-1}v') \\ &= \Phi_l \circ f_{l-1}(g^{-1}u) = (\Phi_g \circ \Phi_l \circ f_{l-1})(u)\end{aligned}$$

Nonlinearity: We define a nonlinearity $\sigma \colon \mathbb{R} \to \mathbb{R}$ and write:

$$(\Phi_\sigma \circ f)(u) = \sigma(f(u)) \tag{16}$$

i.e. ,the nonlinearity is applied on the feature map. The combined operation satisfy Eq.(9):

$$\begin{aligned}(\Phi_\sigma \circ \Phi_g \circ f)(u) &= \sigma(f(g^{-1}u)) \\ &= (\Phi_g \circ \Phi_\sigma \circ f)(u)\end{aligned}$$

Therefore, nonlinear activation is equivariant and does not change the transformation properties of the feature map.

Pooling and subsampling: Similar to standard , we define a pooling for feature map over G as:

$$(\Phi_p \circ f_{l-1})(u) = \max_{k \in uH} f(k) \tag{17}$$

For $H \subset G$. Pooling is equivariant to G:

$$\begin{aligned}(\Phi_p \circ \Phi_g \circ f_{l-1})(u) &= \max_{k \in uH} f_{l-1}(g^{-1}k) \\ &= \max_{k' \in g^{-1}uH} f_{l-1}(k') \\ &= (\Phi_p \circ f_{l-1})(g^{-1}u) = (\Phi_g \circ \Phi_p \circ f_{l-1})(u)\end{aligned}$$

Downsampling $f_l(u)$ require partitioning the group. As discussed previously, the natural way to partitioin a group is by left cosets. We denote the left coset containing u as $[uH]$, and the representive of $[uH]$ by $[u]$. The downsampled output feature map is $f_l \colon \mathcal{Y} \to \mathbb{R}^{N_l}$ where $\mathcal{Y}$ is the homogeneous space of G/H Then, we can define downsampling with respect to H

$$(\Phi_d \circ f_l)(y) = (\Phi_d \circ f_l(u)) = f_l([u]) \tag{18}$$

i.e., output feature map of the left coset containing u has the same value for u belong to the same left coset. This is equivariant to G:

$$\begin{aligned}(\Phi_d \circ \Phi_g \circ f_l)(u) &= f_l(g^{-1}[u]) \\ &= f_l([g^{-1}u]) \\ &= (\Phi_d \circ f_l)(g^{-1}u) \\ &= (\Phi_g \circ \Phi_d \circ f_l)(u)\end{aligned}$$

Furthermore, Downsampled feature map will be invariant to transformation g if the coset contain u is the same as the coset containing $g^{-1}u$.

For example, for plane group $p4$, the group of all four rotations around a lattice points R (point group) are a subgroup of G, Taking $H = R$ for downsampling the quotient group $p4/R$ is isomorphic to the translation of $p4$ and the rotation of the input feature map will have no effect on the output feature map (invariance is a special case of equivariance).

D. Spherical convolution network

Previous section extends the definition of convolution(correlation) to transformations operation of finite group. Now we consider extending it to continuous transformation[10] in three dimensional case.

Correlation on sphere: We consider an image on a unit sphere. We define the image on a sphere as function $f \colon S^2 \to \mathbb{R}^{N_l}$. S^2 is the set of points $x \in \mathbb{R}^3$ on the unit sphere with position specified by (α, β) with $\alpha \in [0, 2\pi]$ and $\beta \in [0, \pi]$. To specify a point at position r within this unit sphere. We can give value $f(|r|)$ to the x on the sphere projected by that point.

Contrary to the image defined on the sphere, rotation of the sphere is given by specified by three parameter: euler angles α, β and γ, $\gamma \in [0, 2\pi]$. They correspond to rotation the north pool of the sphere to an specifid position α, β, followed by a rotation around the axis. In cartesian coordinates, rotation can be specified as a 3×3 matrix.

Using the notation developed above, we define rotated input image as:

$$(\Phi_r \circ f)(x) = f(r^{-1}x) \tag{19}$$

For the first convolution, we have:

$$(\Phi_l \circ f_{l-1})(r) = \int_{s^2} f_{l-1}(x) g_l(r^{-1}x) dx \tag{20}$$

with filter and input feature map defined on S^2. Since the following filter and feature map are defined on group R, We write convolution as:

$$(\Phi_l \circ f_{l-1})(r) = \int_v f_{l-1}(v) g_l(r^{-1}v) du \tag{21}$$

du is the integral element around the rotation r. In practice, the intergral over rotation can be transformed to be $\int\int\int d\alpha d\beta d\gamma$. There equation follows Equation (15) and therefore are equivariant to rotation.

Further simplification can be achieved utilizing the spherical harmonic decomposition. For a function on sphere, it can be decomposed using (real) spherical harmonics:

$$f(x) = \sum_{0 \le l \le b} \sum_{|m| \le l} f_m^l Y_m^l(x) \quad f_m^l = \sum_{S^2} Y_m^l(x) f(x) dx \tag{22}$$

with $Y_m^l(x)$ the spherical harmonics and b the bandwidth of the decomposition. Now, a feature map $f(x)$ is uniquely specified by its decomposition coefficient vector $\mathbf{f}_m^l$.

Similarly, feature map defined on R can be decomposed by Wigner D-function $D^l_{mn}(R)$, with index range $-l \leq m, n \leq l$:

$$f(r) = \sum_{0 \leq l \leq b} (2l+1) \sum_{|m|,|n| \leq l} f^l_{mn} D^l_{mn}(r) \tag{23}$$

. $f(r)$ is equivalently specified by coefficients $\mathbf{f}^l_{mn}$.

R. Kondor and S. Trivedi showed the convolution theorem for convolution on groups, so that Equation (21) can be evaluated on the fourier space.

IV. Steerable CNNs

We describe the steerable CNNs that is purposed by Cohen and Welling 2016[11] and Weiler et al. 2018[12], [13]. Basically, the so called steerable CNN is an extension of the standard CNN that are designed to contain extra (rotational) transformation properties through the filters. Recall that the group equivariant CNNs contain those transformation properties by defining the feature map domain to be that of the group action. For steerable CNNs, the feature map defined is the same as the standard translation CNNs but now the channels are vectors must transform with required transformation properties. In contrast, the channels in the standard CNNs or group equivariant CNNs are independent.

A feature map $\mathcal{F}$ is called steerable with respect to group G of transformation if it has the same transformation properties of the input (we know how to steer the output feature map given a transformed input). One specifically nice property about steerable CNNs is that, since the channels have certain transformation properties, it is very natural to consider the channels as vectors to be transformed by the group action. This properties is suitable for application in physics. However, one of the draw back of the steerable CNNs as purposed is that it applies to data on grids, or volumetric data.

A. Formulation on images

Equivariance: We define an 2D-image: $f_l \colon \mathbb{Z}^2 \to \mathbb{R}^{N_l}$ with N_l channels at layer l. For $x \in \mathbb{Z}^2$, we call $f_l(x)$ a *fiber* of the image, which we consider as a vector in F^l_x vector space. The action of group element $g = tr \in G$ on the input image f is written:

$$[\pi(g)f](x) = f(g^{-1}x) \tag{24}$$

the vector space $\mathcal{F}$ of 2D-image with the linear action $\pi(g)$ form a representation of G

We define a *filter* as a linear map $\Psi \colon \mathcal{F} \to \mathbb{R}^{N_{l+1}}$. These filters can be parameterized just like that in the standard CNN as a tensor with dimension $N_l \times N_{l+1} \times s \times s$, where s are the patch size (for example, 2×2) that the filter operate on, N_l is the input channel and N_{l+1} is the length of the output channel. The filters take a patch of input of the image f_l and all its channels ($N \times s^2$) and output a single vector $f_{l+1}(x) \in F^{l+1}_x$ (fiber vector space). They are in vector space of the representation π and ρ. For a subgroup $H \subset G$, we require the filter Ψ to be *H-equivariant*:

$$\rho(h)[\Psi \circ f] = \Psi \circ \pi(h) f \qquad h \in H \tag{25}$$

recall that ρ gives a representation of H that act on the vector space of fiber F_x. This condition means that, $f_{l+1}(x)$ obtained by applying filter after the transformation of h (by $\pi(h)$) on the image is equivalent to first applying the filter, than apply action of h (by $\rho(h)$) in the vector space of the fiber F^{l+1}_x. Therefore, filter operation is a homomorphism between the two representation for group H. The vector space of all such homomorphism can be denoted as $\mathrm{Hom}_H(\pi, \rho)$.

We apply the filter on feature map by *convoluting it with a translated feature map* (we fix the filter at origin and move the feature map)

$$f_{l+1}(x) = [\Psi \circ f_l](x) = \Psi \star (\pi(x)^{-1} f_l) \tag{26}$$

in $\pi(x)$, x denote a translation, instead of a coordinate. Thene, for a general operation $g = tr$ with $t \in T$, $r \in H$, we have the transformation properties:

$$\begin{aligned}
[\Psi \circ (\pi(tr) f_l)](x) &= \Psi \star \pi(x^{-1}) \pi(tr) f_l \\
&= \Psi \star \pi(x^{-1} tr) f_l \\
&= \Psi \star \pi(r r^{-1} x^{-1} tr) f_l \\
&= \Psi \star \pi(r) \pi(r^{-1} x^{-1} tr) f_l \\
&= \rho(r) \Psi \star \pi \left(\left((tr)^{-1} x \right)^{-1} \right) f_l \\
&= \rho(r) [\Psi \star f_l]((tr)^{-1} x) \\
&= \rho(r) (\pi(tr) f_{l+1})(x)
\end{aligned}$$

$[\Psi \circ (\pi(tr) f_l)]$ means we transform the input feature map f_l by the action of tr, then applying the filter. The right hand side $\rho(r)[\Psi \star f_l]((tr)^{-1}x)$ means we 1) apply the filter, 2) transform the feature map using relation $x \to (tr)^{-1}x$ and 3) apply the transformation $\rho(r)$ on the output fibers. If we define transformation Π (induced transformation) on the output feature map:

$$[\Pi(tr) f](x) = \rho(r)(\pi(tr) f)(x) \tag{27}$$

Then we obtain the equivariance:

$$\Psi \circ \pi(g) f = \Pi(g)[\Psi \circ f] \qquad \text{(Equivariance)}$$

we take care to distinguish the effect of π and Π. π operation on the feature map as according to

$$[\pi(g) f](x) = f(g^{-1}x)$$

which trans-rotate the feature map, while Π trans-rotate the feature map, as well as rotating the fibers.

V. Formulation for 3-dimensional space

In Weiler et al. 2018, a steerable CNN for SE(3) is purposed. The starting point of the formulation is the induced representation equation (27), which gives the transformation properties for a group action g on a feature map f:

$$[\Pi(tr) f](x) = \rho(r)(\pi(tr) f)(x) \tag{28}$$

Writing for a general kernel $\kappa \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{N_{l+1} \times N_l}$, the later part correspond to weight matrix between N_l input feature channels and N_{l+1} output feature channels. we denote the mapping of feature $\mathcal{F}_n \to \mathcal{F}_{n+1}$ as

$$f_{l+1}(x) = [\kappa \circ f_l](x) = \int_{\mathbb{R}^3} \kappa(x, y) f_l(y) dy \tag{29}$$

This is a equivariant mapping to SE(3) if:

$$\kappa \circ [\Pi_1(g)f](x) = \Pi_2(g)[\kappa \circ f](x) \tag{30}$$

The left hand side of the equation can be evaulated as:

$$\begin{aligned}[\kappa \circ \Pi_1(tr)f](x) &= [\kappa \circ \rho(r)(\pi(tr)f)](x) \\ &= \int_{\mathbb{R}^3} \kappa(x,y)\rho_1(r)\pi(tr)f(y)dy \\ &= \int_{\mathbb{R}^3} \kappa(x,y)\rho_1(r)f((tr)^{-1}y)dy \\ &= \int_{\mathbb{R}^3} \kappa(x,(tr)y)\rho_1(r)f(y)dy\end{aligned}$$

On the right, we have:

$$\begin{aligned}\Pi_2(g)[\kappa \circ f](x) &= \Pi_2(g)\int_{\mathbb{R}^3} \kappa(x,y)f(y)dy \\ &= \rho_2(r)\int_{\mathbb{R}^3} \kappa((tr)^{-1}x,y)f(y)dy\end{aligned}$$

Equating the two parts and using $g = tr$, we therefore require:

$$\begin{aligned}\kappa(x,gy)\rho_1(r) &= \rho_2(r)\kappa(g^{-1}x,y) \\ \kappa(gx,gy)\rho_1(r) &= \rho_2(r)\kappa(x,y) \\ \kappa(gx,gy) &= \rho_2(r)\kappa(x,y)\rho_1(r)^{-1}\end{aligned} \tag{31}$$

Therefore, to satisfy the equivariance equation (30), the kernel need to behave as above with respect to ρ_1 and ρ_2.

For a kernel in this form, it is translation invariant: if g is a pure translation, then $\kappa(gx,gy) = \kappa(x,y)$. Therefore, taking $g = x^{-1}$, we find:

$$\kappa(0, y-x) = \kappa(x,y) \tag{32}$$

i.e., κ only depend on the relative distance between its inputs and equation (29) reduce to a standard correlation (taking $g = y^{-1}$ gives convolution):

$$f_{l+1}(x) = [\kappa \circ f](x) = \int_{\mathbb{R}^3} \kappa(y-x)f(y)dy \tag{33}$$

where the kernel κ is a function $\kappa\colon \mathbb{R}^3 \to \mathbb{R}^{N_{l+1}\times N_l}$

A. Rotation equivariant

In Weiler et al. 2018, rotational equivariant version of the steerable CNN is applied to image data and achieve good accuracy. This is named steerable CNN but in concept, this work is closer to the group equivariant CNNs rather than the above mention steerable CNNs. The key point in their work is that, just like in standard CNN where image is convoluted with weight sharing translated filters, they convolute the image with rotated version of the filter with weight sharing. This is achieved by writing the filter as a combination of function:

$$\begin{aligned}\Psi(x) &= \sum_j \sum_k w_{jk}\psi_{jk}(x) \\ &= \sum_j \sum_k w_{jk}R_j(r)e^{ik\phi}\end{aligned}$$

r and ϕ are the polar coordinates of a pixel and R_j a radial basis function. In this form, a rotation of the filter is given by simply $\rho_\theta e^{ikx} = e^{ik(x-\theta)}$ We convolute the image with the image I with N rotated filter $\rho_\theta\Psi(x)$ ($\theta = 2\pi/N$):

$$\begin{aligned}y_{c'}(x,\theta) &= \sum_c [I_c \circ \rho_\theta\Psi](x) \\ &= \sum_c \sum_j \sum_k w_{c'cjk}e^{-ik\theta}[I_c \circ \psi_{jk}](x)\end{aligned}$$

We can recognize that the network operate very similar to spherical CNN (Cohen et al. 2018) with decretized rotations. The channels are independent and the notion of fiber are not presented.

VI. Euclidean neural network

A. Euclidean neural network

Euclidean nueral network purposed by Thomas et al. 2018[14] is different from the previous network in that *The euclidean neural network operate on points throughout the structure: convolution is performed on each point with respect to other points.* Equivariance is achieved by their definition of point convoultion, but is simpler in the form because the discreteness of the points.

At each layer, each point a in the point cloud are associated with a vector $(\mathbf{r}_a, \mathbf{x}_a)$ in the vector space $R^3 \oplus \mathcal{X}$, where $\oplus$ indicate direct sum. We require that vector space $\mathcal{X}$ are span by the spherical harmonics Y_m^l for $0 \le l \le b$ and $|m| \le l$. We denote such a feature vector as $V_a cm^l$, where c denotes channels (depth). Under a rotation R around a chosen origin, The input feature thus transforms as:

$$(\mathbf{r}_a, V^l_{acm}) \to (R\mathbf{r}_a, \sum_{m'} D^l_{mm'}(R)V^l_{acm'}) \tag{34}$$

We define a filter to be of the following form:

$$F^{l_f,l_i}_{cm_f}(\mathbf{r}) = R^{l_f,l_i}_c(r)Y^{l_f}_{m_f}(\mathbf{r}) \tag{35}$$

with $r = |\mathbf{r}|$, and $R^{l_f,l_i}_c(r)$ are the parameter of the filter function that can be learned. The filter functions are equivariant to rotation:

$$F^{l_f,l_i}_{cm_f}(R\mathbf{r}) = R^{l_f,l_i}_c(r)Y^{l_f}_{m_f}(R\mathbf{r}) \tag{36}$$

$$= R^{l_f,l_i}_c(r)\sum_{m'_f} D^{l_f}_{m_f m'_f}(R)Y^{l_f}_{m_f}(\mathbf{r}) \tag{37}$$

In the following, we will first define the components of the network. Then we will show that they are equivariant to rotation. As well as invariant to translation and permutation.

Point convolution

we define a convolution layer as:

$$\begin{aligned}V^{l_o}_{acm_o} &= L^{l_o}_{acm_o}(\mathbf{r}, V^{l_i}_{cm_i}) \\ &= \sum_{m_f,m_i} C^{l_o,m_o}_{(l_f,m_f)(l_i,m_i)} \sum_{b\in S} F^{l_f,l_i}_{cm_f}(\mathbf{r}_{ab})V^{l_i}_{bcm_i}\end{aligned} \tag{38}$$

where l_i and l_f is the rotation order of the input and the filter, $m = -l_f, \ldots, l_f$ is the index of the basis function. $C^{l_o,m_o}_{(l_f,m_f)(l_i,m_i)}$ is the *Clebsch-Gordan coefficients* Note that we do not contain index a in

$$L^{l_o}_{acm_o}(\mathbf{r}, V^{l_i}_{cm_i})$$

because the output depend on the feature vector $(\mathbf{r}, V^{l_i}_{cm_i})$ from all points in the set S. Note that we can understand the 'convolution' as the convolution between the filter function and a delta function marking the position of the point, at $\mathbf{r}_{ab}$.

Since the output only depend on relative distance $\mathbf{r}_{ab} = \mathbf{r}_a - \mathbf{r}_b$, the output is invariant under a translation of all points. Also, the summation in Equation (38) means the result is invariant under permutation of points.

We can show that the above point convolution is equivariant to rotation (Appendix A): rotation on the input feature space (l_i, m_i) correspond to rotation on output feature space (l_o, m_o).

Self-interaction

We define a self-interaction between the feature at each point a as:

$$V^l_{acm} = \sum_{c'} W^l_{cc'} V^l_{ac'm} \tag{39}$$

Such operation is independent on permutation and translation. It is also equivariant to rotation:

$$\begin{aligned}[W \circ R](V^l_{acm}) &= \sum_{c'} W^l_{cc'} \sum_{m'} D^l_{mm'}(R) V^l_{acm'} \\ &= \sum_{m'} D^l_{mm'}(R) \sum_{c'} W^l_{cc'} V^l_{acm'} \\ &= [R \circ W](V^l_{acm})\end{aligned}$$

However, it should be noted that the self-interaction coefficients W are used for every m. If for different m we have different weight, we than have:

$$\begin{aligned}[W \circ R](V^l_{acm}) &= \sum_{c'} W^{lm}_{cc'} \sum_{m'} D^l_{mm'}(R) V^l_{acm'} \\ &= \sum_{m'} D^l_{mm'}(R) \sum_{c'} W^{lm}_{cc'} V^l_{acm'}\end{aligned}$$

which is different from $\sum_{c'} W^{lm'}_{cc'} V^l_{acm'}$ by first applying self-interaction.

Nonlinearity

For nonlinearity, we can use:

$$\begin{aligned} &h^l(V^0_{ac} + b^0_c) && \text{for } l = 0 \\ &h^l(\|V\|^l_{ac} + b^l_c) V^l_{acm} && \text{for } l > 0\end{aligned}$$

with $h\colon \mathbb{R} \to \mathbb{R}$ and

$$\|V\|^l_{ac} = \sqrt{\sum_m |V^l_{acm}|^2}$$

Nonlinearity is invariant for permutation and translation. For rotation, because rotation does not change length in the vector space, $\|V\|^l_{ac}$ is therefore invariant of rotation:

$$\|D(R)V\| = \|V\| \tag{40}$$

Thus, $h^l(\|V\|^l_{ac} + b^l_c)$ is invariant with respect to rotation. The rotational properties is the output is solely given by V^l_{acm}. Therefore, nonlinearity defined by Equation (**??**) is equivariant with respect to rotation.

Relationship with spherical CNNs

Here, we show the euclidean neural network can be understood as a special case of the more general spherical CNNs with fixed filter. We first consider Equation (38) with $l_i = m_i = 0$, i.e., only scalars on each point. Omitting the channel index c, equation (38) becomes:

$$\begin{aligned}V^l_{am} &= \sum_{b \in S} F^l_m(\mathbf{r}_{ab}) V_b \\ &= \sum_{b \in S} \left[R^l(|r|) V_b\right] Y^l_m(\mathbf{x})\end{aligned} \tag{41}$$

The part in square bracket is scalar value. This activation can be achieved by convoluting a filter with

$$f(\mathbf{r}) = \sum_{b \in S} \left[R^l(|r|) V_b\right] \delta(\mathbf{x}_{ab}) \tag{42}$$

$\delta(\mathbf{x}_{ab})$ is zero unless the projection of the point b fall on the sphere at $\mathbf{x}_{ab}$. We define the filter as:

$$g_l(x) = \sum_{0 \le l \le b} \sum_{|m| \le l} Y^l_m(x) \tag{43}$$

Putting the above form into Equation (20):

$$\begin{aligned}V_a(r) &= \int_{s^2} \sum_{b \in S} \left[R^l(|r|) V_b\right] \delta(\mathbf{x}_{ab}) \sum_{l,m} Y^l_m(r^{-1}x) dx \\ &= \sum_{l,m} \sum_{b \in S} \left[R^l(|r|) V_b\right] Y^l_m(r^{-1}\mathbf{x}_{ab}) \\ &= \sum_{l,m} \sum_{b \in S} \left[R^l(|r|) V_b\right] \sum_n D^l_{mn}(r^{-1}) Y^l_n(\mathbf{x}_{ab}) \\ &= \sum_{l,m,n} D^l_{mn}(r^{-1}) \sum_{b \in S} \left[R^l(|r|) V_b\right] Y^l_n(\mathbf{x}_{ab})\end{aligned}$$

The coefficients

$$V^l_{an} = \sum_{b \in S} \left[R^l(|r|) V_b\right] Y^l_n(\mathbf{x}_{ab})$$

therefore uniquely define the feature map on r. It describes the local environment of the points equivariantly to rotation r.

If the point b are associated with $l > 0$, then we also need to take into account the rotational property of these points. This is achieved using the properties of Clebsch-Gordan coefficients, similar to the addition of angular moment in quantum mechanics.

VII. Applicaton

In this section, we show that the rotational equivariant network is able to extract vector information on atoms, from the crystal structures.

VIII. Conclusion

We reviewd recent works that operate on molecular and crystal structures. Basic ideal of translational and rotational equivariance is introduced that form the basis for the development of rotational equivariant networks. Furthermore, we showed an application of the network.

Reference

[1] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," en, *arXiv:1706.02413 [cs]*, Jun. 2017, arXiv: 1706.02413. [Online]. Available: http://arxiv.org/abs/1706.02413 (visited on 02/04/2022).

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," en, *arXiv:1612.00593 [cs]*, Apr. 2017, arXiv: 1612.00593. [Online]. Available: http://arxiv.org/abs/1612.00593 (visited on 02/04/2022).

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," en, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021, arXiv: 1901.00596, ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2020.2978386. [Online]. Available: http://arxiv.org/abs/1901.00596 (visited on 02/17/2022).

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," en, *arXiv:1704.01212 [cs]*, Jun. 2017, arXiv: 1704.01212. [Online]. Available: http://arxiv.org/abs/1704.01212 (visited on 02/15/2022).

[5] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," en, *Nature Communications*, vol. 8, no. 1, p. 13 890, Apr. 2017, ISSN: 2041-1723. DOI: 10.1038/ncomms13890. [Online]. Available: http://www.nature.com/articles/ncomms13890 (visited on 12/17/2021).

[6] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet – A deep learning architecture for molecules and materials," en, *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241 722, Jun. 2018, ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.5019779. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.5019779 (visited on 12/17/2021).

[7] J. Klicpera, J. Groß, and S. Günnemann, "Directional Message Passing for Molecular Graphs," en, *arXiv:2003.03123 [physics, stat]*, Mar. 2020, arXiv: 2003.03123. [Online]. Available: http://arxiv.org/abs/2003.03123 (visited on 01/25/2022).

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 978-0-262-03561-3.

[9] E. Kauderer-Abrams, "Quantifying Translation-Invariance in Convolutional Neural Networks," en, *arXiv:1801.01450 [cs]*, Dec. 2017, arXiv: 1801.01450. [Online]. Available: http://arxiv.org/abs/1801.01450 (visited on 02/02/2022).

[10] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," en, *arXiv:1801.10130 [cs, stat]*, Feb. 2018, arXiv: 1801.10130. [Online]. Available: http://arxiv.org/abs/1801.10130 (visited on 01/25/2022).

[11] T. S. Cohen and M. Welling, "Steerable CNNs," en, *arXiv:1612.08498 [cs, stat]*, Dec. 2016, arXiv: 1612.08498. [Online]. Available: http://arxiv.org/abs/1612.08498 (visited on 02/04/2022).

[12] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen, "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data," en, *arXiv:1807.02547 [cs, stat]*, Oct. 2018, arXiv: 1807.02547. [Online]. Available: http://arxiv.org/abs/1807.02547 (visited on 01/31/2022).

[13] M. Weiler, F. A. Hamprecht, and M. Storath, "Learning Steerable Filters for Rotation Equivariant CNNs," en, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 849–858, ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00095. [Online]. Available: https://ieeexplore.ieee.org/document/8578193/ (visited on 02/08/2022).

[14] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds," en, *arXiv:1802.08219 [cs]*, May 2018, arXiv: 1802.08219. [Online]. Available: http://arxiv.org/abs/1802.08219 (visited on 11/30/2021).

APPENDIX A
EQUIVARIANCE OF POINT CONVOLUTION

Here we show that the point convolution introduced as Equation (38) is indeed rotational equivariant. Recall that the point convolution is given as:

$$\begin{aligned} V^{l_o}_{acm_o} &= L^{l_o}_{acm_o}(\mathbf{r}, V^{l_i}_{cm_i}) \\ &= \sum_{m_f, m_i} C^{l_o, m_o}_{(l_f, m_f)(l_i, m_i)} \sum_{b \in S} F^{l_f, l_i}_{cm_f}(\mathbf{r}_{ab}) V^{l_i}_{bcm_i} \end{aligned}$$

Under transformation (Equation (34)), we have:

$$\begin{aligned} [L \circ R](\mathbf{r}, V^{l_i}_{cm_i}) &= L^{l_o}_{acm_o}(R\mathbf{r}, \sum_{m'_i} D^{l_i}_{m_i m'_i}(R) V^{l_i}_{cm'_i}) \\ &= \sum_{m_f, m_i} C^{l_o, m_o}_{(l_f, m_f)(l_i, m_i)} \sum_{b \in S} F^{l_f, l_i}_{cm_f}(R\mathbf{r}_{ab}) \sum_{m'_i} D^{l_i}_{m_i m'_i}(R) V^{l_i}_{bcm'_i} \\ &= \sum_{m_f, m_i} C^{l_o, m_o}_{(l_f, m_f)(l_i, m_i)} \sum_{b \in S} R^{l_f, l_i}_c(r) \sum_{m'_f} D^{l_f}_{m_f m'_f}(R) Y^{l_f}_{m_f}(\mathbf{r}_{ab}) \sum_{m'_i} D^{l_i}_{m_i m'_i}(R) V^{l_i}_{bcm'_i} \\ &= \sum_{m'_o} D^{l_o}_{m_o m'_o}(R) \sum_{m_f, m_i} C^{l_o, m'_o}_{(l_f, m_f)(l_i, m_i)} \sum_{b \in S} F^{l_f, l_i}_{cm_f}(\mathbf{r}_{ab}) V^{l_i}_{bcm_i} \\ &= \sum_{m'_o} D^{l_o}_{m_o m'_o}(R) L^{l_o}_{acm'_o}(\mathbf{r}, V^{l_i}_{cm_i}) = [R \circ L](\mathbf{r}, V^{l_i}_{cm_i}) \end{aligned}$$

which shows that point convolution is equivariant. We used the properties of Clebsch-Gordan coefficients:

$$\sum_{m_f m_i} C^{l_o, m_o}_{(l_f, m_f)(l_i, m_i)} D^{l_f}_{m_f m'_f}(R) D^{l_i}_{m_i m'_i}(R) = \sum_{m'_o} D^{l_o}_{m_o m'_o}(R) C^{l_o, m'_o}_{(l_f, m'_f)(l_i, m'_i)} \tag{44}$$