An Introduction to Equivariant Convolutional Neural Networks for Continuous Groups

Lecture Notes

dr.ir. Erik J. Bekkers

August 2021

Contents

Part I
Regular Group Convolutions

1 Introduction to convolutions

1.1 Notations

First let us get acquainted with the notation used in these notes. We primarily consider the construction of neural networks the processing of real-valued signal data. As such our primary objects of interest our multi-channel functions $\underline{f} : X \to \mathbb{R}^{N_c}$ defined on some domain X and with N_c the number of channels. We furthermore assume that these signals are square integrable as we will often rely on a notion of inner products[1]. Let $\mathbb{L}_2(X)^{N_c}$ denote the space of square integrable signals. We will often limit our analyses to scalar valued signals, i.e., for which $N_c = 1$. Scalar valued signals will be denoted with $f : X \to \mathbb{R}$ where we omit the underline. We can then denote the signal value of a multi-channel signal at a location $x \in X$ as a vector $\underline{f}(x) = (f_1(x), \dots, f_{N_c}(x))^T \in \mathbb{R}^{N_c}$.

For mathematical convenience we will often consider signals as functions on continuous domain, however, note that in practice data almost always comes sampled on a finite discrete grid. We will then denote a discretization of X with $\mathbf{X}$. Our analyses mostly take place in the continuous settings and discretization aspects will be treated separately. For now, we often abstractly represent the domain of a signal with X as we will later consider several cases for it; X could be $\mathbb{R}^d$, a group, or a homogenous space of a group.

Example 1.1 (Signals, Dense signals, Images, Volumetric data)**.** We will assume that signal data such as time series, images and volumetric data are sampled instantiations of continuous signals. E.g., an 28×28 RGB image is a continuous signal sampled on a discrete pixel grid $\mathbf{X} = \{0, 1, \dots, 27\} \times \{0, 1, \dots, 27\} \subset \mathbb{R}^2$ that assigns to each discrete pixel location $\mathbf{x} = (x, y) \in \mathbf{X}$ an RGB value $\underline{f}(\mathbf{x}) \in \mathbb{R}^3$. For mathematical convenience we will however often treat signal data as continuous functions and treat the discretization aspect and numerical aspects separately.

Example 1.2 (Sparse signals, Point clouds)**.** Mathematically we treat point clouds little differently from dense data and assume that the underlying signal is continuous but is simply measured on a finite set of pixel locations $\mathbf{X} = \{x_1, x_2, \dots, x_{N_x}\} \subset X$. A sampled point cloud signal then consists of position-value pairs $(x_i, \underline{f}(x_i))$, where we may decide to write $\mathbf{f}_i := \underline{f}(x_i)$.

We will denote vectors in $\mathbb{R}^d$ with boldface symbols such as e.g. a coordinate vector $\mathbf{x} \in \mathbb{R}^d$ or a feature vector $\mathbf{f} \in \mathbb{R}^{N_c}$. Note that a multi-channel, or vector valued, signal $\underline{f}$ is not a standard Euclidean vector. It will therefore not be boldfaced but we will underline it. Its value sampled at a location $\mathbf{x}_i$ is however a vector, hence we will occasionally use the notation $\mathbf{f}_i := \underline{f}(\mathbf{x}_i)$ as in Example 1.2.

[1] We will not concern ourselves with the analysis of function spaces but simply use notations such as $\mathbb{L}_2(X)$ to conveniently denote a space of functions even when we occasionally only require $\mathbb{L}_1(X)$ or when it could come from other spaces. When necessary we will make more accurate specifications.

1.2 Convolution and cross-correlation

Before we define group convolutions let us first revisit the definition of the convolution operator on $\mathbb{R}^d$ and work a bit on the intuition for why it is such a successful building block to build deep leanring architectures.

Definition 1.1 ($\mathbb{R}^d$-Convolution)**.** The convolution operator is denoted with $*$ and a convolution between two signals $k, f \in \mathbb{L}_2(\mathbb{R}^d)$ is denoted with $f * g$. The convolution between two signals is again a signal that is defined by

$$(k * f)(\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x} - \tilde{\mathbf{x}}) f(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \,. \tag{1}$$

Its definition for multi-channel signals $\underline{k}, \underline{f} \in \mathbb{L}_2(\mathbb{R}^d)^{N_c}$ is given by

$$(\underline{k} * \underline{f})(\mathbf{x}) = \sum_{c}^{N_c} \int_{\mathbb{R}^d} k_c(\mathbf{x} - \tilde{\mathbf{x}}) f_c(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \,. \tag{2}$$

The first input to the convolution (k or $\underline{k}$) is called the convolution kernel.

Definition 1.2 ($\mathbb{R}^d$-Correlation)**.** Cross-correlation operator, or simply correlation operator, is denoted with $\star$ and a correlation between two signals $k, f \in \mathbb{L}_2(\mathbb{R}^d)$ is denoted with $f * g$ and defined by

$$(k \star f)(\mathbf{x}) = \int_{\mathbb{R}^d} k(\tilde{\mathbf{x}} - \mathbf{x}) f(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \,. \tag{3}$$

Similarly for multi-channel signals $\underline{k}, \underline{f} \in \mathbb{L}_2(\mathbb{R}^d)^{N_c}$ it is defined by

$$(\underline{k} \star \underline{f})(\mathbf{x}) = \sum_{c}^{N_c} \int_{\mathbb{R}^d} k_c(\tilde{\mathbf{x}} - \mathbf{x}) f_c(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \,. \tag{4}$$

The first input to the correlation (k or $\underline{k}$) is called the correlation kernel.

Correlations are convolutions with reflected kernels The definition of convolution and cross-correlation is rather similar and from a deep learning perspective are equally useful. Namely, convolutions are equal to correlations simply via a kernel reflection, i.e., $k * f = \check{k} \star f$ where $\check{k}(\mathbf{x}) = k(-\mathbf{x})$. Then, since the kernels k are learned anyway, it does not matter whether a layer in a neural network is parametrized via correlations or convolutions. Although technically not very precise, in deep learning literature the term convolution is often used for any of the equations (1-4).

From a mathematical point of view the convolution operator may be preferable as it has the property that it commutes with swapping the functions, i.e., $k * f = f * k$. This is not the case for correlations. Nevertheless, we prefer to work with the correlation operator in these lecture notes as it allows for a more

intuitive introduction to group correlations (or convolutions) by the notion of template matching. This is what we discuss next.

Exercise 1.1. Verify that correlations relate to convolutions via a kernel reflection.

Exercise 1.2. Verify that convolutions commute ($k \star f = f \star k$) and correlations do not.

1.3 Convolutions/correlations seen as template matching

Convolutions and correlations are most intuitively thought of as template matching, in which the kernel k acts as a template that is moved over the underlying signal f and matched at each location $\mathbf{x}$. The matching is then done in terms of an inner product between the translated kernel $k(\cdot - \mathbf{x})$ and the underlying signal $f(\cdot)$ in terms of inner products. Let us formalize the definition of the inner product and the norm that it induces.

Definition 1.3 ($\mathbb{L}_2(\mathbb{R}^d)$-inner product)**.** The $\mathbb{L}_2$-inner product between two square integrable (with respect to measure $\mathrm{d}\tilde{x}$) signals $k, f \in \mathbb{L}_2(X)$ or multi-channel signals $\underline{k}, \underline{f} \in \mathbb{L}_2(X)^{N_c}$ are both denoted with $(\cdot,\cdot)_{\mathbb{L}_2(X)}$ and given by

$$(k, f)_{\mathbb{L}_2(X)} = \int_X k(\tilde{x}) f(\tilde{x}) \mathrm{d}\tilde{x}\,, \tag{5}$$

$$(\underline{k}, \underline{f})_{\mathbb{L}_2(X)} = \sum_c^{N_c} \int_X k_c(\tilde{x}) f_c(\tilde{x}) \mathrm{d}\tilde{x}\,. \tag{6}$$

Definition 1.4 ($\mathbb{L}_2(\mathbb{R}^d)$-norm)**.** Given the definition of the inner products in (5-6) the $\mathbb{L}_2(X)$ norms for scalar and multi-channel signals are both denoted with $\|\cdot\|_{\mathbb{L}_2(X)}$ and given by

$$\|f\|^2_{(\mathbb{L}_2(X)} = (f, f)_{\mathbb{L}_2(X)}\,, \tag{7}$$

$$\|\underline{f}\|^2_{(\mathbb{L}_2(X)} = (\underline{f}, \underline{f})_{\mathbb{L}_2(X)}\,. \tag{8}$$

Exercise 1.3. Verify that the inner product can interpreted as a similarity measure. Consider the case of $k, f \in \mathbb{L}_2(\mathbb{R}^d)$ with $\|k\|_{(\mathbb{L}_2(\mathbb{R}^d)} = \|f\|_{(\mathbb{L}_2(\mathbb{R}^d)} = 1$. Show that $(k, f)_{\mathbb{L}_2(\mathbb{R}^d)} = 1$ when $k = f$ and that $(k, f)_{\mathbb{L}_2(\mathbb{R}^d)} = -1$ when $k = -f$.

Intuition, the inner product as a similarity measure The intuition on $\mathbb{L}_2(X)$-inner products is similar to that of what you may be used to on the standard inner product with Euclidean vectors. Recall that functions are also vectors; namely infinite dimensional vectors whose "components" $f(x)$ are indexed with infinitely many $x \in X$. Also for functions there is a notion of alignment ($(k, f)_{\mathbb{L}_2(X)} > 0$) or orthogonality ($(k, f)_{\mathbb{L}_2(X)} = 0$) and when dividing the inner product of Eq. 5 with the norms of the functions one arrives at

a norm-independent similarity measure that is the functional equivalent of the well-known cosine-similarity. Template matching with such normalized inner product is called normalied cross-correlation and can for example be useful to detect patterns in image data invariantly to local brightness and crontast variations, see e.g. [Bekkers et al., 2015] where this is done in a group convolutional setting.

Template matching Let us consider template matching using the inner product as matching score, and define $\mathcal{T}_{\mathbf{x}}$ the translation operator as

$$[\mathcal{T}_{\mathbf{x}}k](\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}} - \mathbf{x})\,. \tag{9}$$

hen cross-correlation is no different than matching shifted kernels with the underlying signal for all possible translations $\mathbf{x}$ via

$$(k \star f)(\mathbf{x}) = (\mathcal{T}_{\mathbf{x}}k, f)_{\mathbb{L}_2(\mathbb{R}^d)}\,. \tag{10}$$

This is precisely how one typically looks at the convolution layers in a convolutional neural network. In each layer the input feature map is correlated with a set of kernels that are responsible to detect local patterns or features. For example, when following the template matching with a ReLU activation function only the positive responses survive, those are the locations where the kernel aligned with the background. The interpretation of the output feature maps is then an assignment of a score for the presence of a feature at each position.

Weight sharing in CNNs The patterns that are learned in CNNs are usually highly redundant when it comes to geometric transformations such as rotations. One often sees that in the earlier layers of a CNN the kernels are rotated copies of each other. E.g., one kernel may act as edge detector in one direction, and one might act as edge detector in another location. The power of CNNs is that it allows for weight sharing over different locations in the data; a kernel that is shared over all positions through the correlation operator. However, there appears to be no weight sharing over rotations/orientations while inspection of learned features show that this may significanlty decrease redundancy. In many datastructures one can expect basic patterns to appear under arbitrary rotations, such as edges, corners, lines, etc. in 2D images or 3D medical image data or certain configuraitons of clusters of atoms in molecular data.

Weight sharing in G-CNNs Group convolutional neural networks allow for weight sharing over a larger group of transformations than just translations. The principle is exactly the same as in classical CNNs. Namely, through template matching with correlation kernels, however, now the kernel is not just translated but could e.g. be translated, rotated, and/or scaled. To formalize this approach we will soon introduce the appropriate group theoretical prerequisites, but the intuition is as follows.

1.4 Group correlations

Group correlations Consider a transformation operator $\mathcal{L}_g$ that is parameterized by the transformation paramaters g. This could e.g. be a roto-translation parametrized by $g = (\mathbf{x}, \theta)$ with $\mathbf{x}$ the translation vector and θ the rotation angle, where $\mathcal{L}_{(\mathbf{x},\theta)}$ is defined as

$$[\mathcal{L}_{(\mathbf{x},\theta)}k](\mathbf{x}) = k(\mathbf{R}_\theta^{-1}(\tilde{\mathbf{x}} - \mathbf{x}))\,. \tag{11}$$

Then a roto-translation lifting correlation is exactly the same as before (Eq. 10), but with a different transformation applied to the kernel:

$$\begin{aligned}(k \star_{SE(2)} f)(\mathbf{x}, \theta) &= (\mathcal{L}_{(\mathbf{x},\theta)}k, f)_{\mathbb{L}_2(\mathbb{R}^d)} \\ &= \int_{\mathbb{R}^d} k(\mathbf{R}_\theta^{-1}(\tilde{\mathbf{x}} - \mathbf{x}))f(\tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}}\,.\end{aligned} \tag{12}$$

It is a matching of a roto-translated kernel with the underlying signal. Note, however, that the output is no longer a feature map on $\mathbb{R}^d$ but a higher dimensional feature map on the space of transformation parameters. I.e., let $\underline{f}^{l+1}$ denote such an output feature map, then $\underline{f}^{l+1} \in \mathbb{L}_2(SE(2))^{N_c}$ where $SE(2)$ denotes the roto-translation group, or special Euclidean motion group.

G-CNNs G-CNNs then are NN architectures in which feature maps are transformed by consistent application of group correlations of the form given in Eq. (10 and Eq. (12). This means that after the first group convolution, after which a feature map on $SE(2)$ is obtained, the layers are parametrized by three-dimensional kernels that characterize patterns of position-orientation localized features[2] and which are transformed by the application of $\mathcal{L}_g$ that now transforms the $SE(2)$ kernels rather than 2D kernels. This requires a new definition of $\mathcal{L}_g$ as so far we have only defined how it transforms 2D signals. Instead of writing out what $\mathcal{L}_g$ looks like every time we need it, it is common to threat G-CNN on a high level in terms of group theory. We will do this in the upcoming chapters. Once familiar with group theory one can keep the equations simple (though initially seemingly abstract) and construct operators/layers in a principled way.

[2]Where a 2D kernel can be though of as assigning expected values to relative positions, a $SE(2)$ kernel assigns values to relative positions and rotations, relative to a central pose. E.g. in terms of a face detector, the kernel would like to see two eyes, two ears and a mouth at positions and orientations relative to the position and orientation of a central point, e.g. the nose.

2 Mathematical tools: basic group theory

In this chapter we will cover the very basics of group theory. Covering the notions of a group structure (product, inverse, identity), representations and equivariance. This will be sufficient to understand G-CNNs and implement regular group convolutional neural networks. Later, in the second part of these lecture notes in Part II we will expand the group theoretical tool set with concepts that allow us to build steerable G-CNNs. Let us start with the beginning.

2.1 Groups

Definition 2.1 (Group, group product, group inverse)**.** A group is an algebraic structure that consists of a set G and a binary operator $\cdot$ called the group product, that satisfies the following axioms:

- *Closure*: for all $h, g \in G$ we have $h \cdot g \in G$;
- *Identity*: there exists an identity element $e \in G$;
- *Inverse*: for each $g \in G$ there exists an inverse element $g^{-1} \in G$ such that $g^{-1} \cdot g = g \cdot g^{-1} = e$; and
- *Associativity*: for each $g, h, i \in G$ we have $(g \cdot h) \cdot i = g \cdot (h \cdot i)$.

Transformation groups As we will see we can get more intuition about groups when thinking about them as representing transformations, however, to get started it may be sufficient to think of groups purely in an algebraic sense; like for vector spaces there is a formal definition that involves operators that can be used to manipulate their elements. E.g. a vector space is set of elements for which we have a notion of vector addition and scalar multiplication. For groups we work with a different algebraic structure in which the group product plays a central role. In these lecture notes we most regularly treat groups as transformation groups[3], in which the group product can be interpreted as a rule for computing the net effect of combining two transformations and the inverse as a rule for obtaining the transformation parameters that would undo the original one.

[3]Though group structures appear in many places, e.g., as symmetries in particle physics

Example 2.1 (Translation group $G = (\mathbb{R}^d, +)$)**.** The translation group consists of the set of translation vectors $\mathbb{R}^d$ and is equiped with the group product and inverse given by

$$g \cdot \tilde{g} = (\mathbf{x} + \tilde{\mathbf{x}}) \,, \tag{13}$$

$$g^{-1} = (-\mathbf{x}) \,, \tag{14}$$

with group elements $g = (\mathbf{x}), \tilde{g} = (\tilde{\mathbf{x}}) \in \mathbb{R}^d$. As a group, it is referred to with $(\mathbb{R}^d, +)$ rather than $\mathbb{R}^d$ as to avoid confusion on whether or not the elements should be regarded as group elements or vectors. The group product can be interpreted as the the computational rule for obtaining the parameters that describe the net effect of first applying a translation with parameters $\tilde{g} = (\tilde{\mathbf{x}})$ followed by a translation by $g = (\mathbf{x})$.

Note that in Example 2.1 we do not directly represent the group elements as the vectors $\mathbf{x}, \tilde{\mathbf{x}}$ but rather write $g, \tilde{g}$ to indicate that we are talking about group elements. We do this as to avoid confusion on whether or not we should treat the elements as group elements or vectors. That is, the group product and inverse are defined for group elements and not vectors (what would be the inverse of a vector $\mathbf{x}^{-1}$?). Conversely, denoting group elements in vector notation implies that it is a vector for which a scalar multiplication exists, however, for group elements there is no notion of a scalar product.

Lie groups In these notes, we will primarily focus on continuous transformation groups such as translations, rotations and scalings. When such a continuous group also has the structure of a manifold it is called a Lie group. Although Lie groups have additional structure that can be leveraged to build equivariant deep learning algorithms [Bekkers, 2019, Finzi et al., 2020, Hutchinson et al., 2021, Dehmamy et al., 2021, Finzi et al., 2021], we will postpone a deeper discussion to Part **??**.

Definition 2.2 (Lie group)**.** A Lie group is a continuous group that is also a differentiable manifold.

Example 2.2 (Rotation group, special orthogonal group $G = SO(d)$)**.** The rotation group, also known as the special orthogonal[a] group, is a Lie group that consists of the set of $d \times d$-matrices with determinant 1, i.e., the set of rotation matrices $\{\mathbf{A} \in \mathbb{A}^{d \times d} \mid \mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = 1 \wedge \det \mathbf{A} = 1\}$, together with the group product and inverse given by the matrix product and inverse respectively:

$$g \cdot g' := (\mathbf{R}\mathbf{R}') \,, \tag{15}$$

$$g^{-1} := (\mathbf{R}^{-1}) \,. \tag{16}$$

with $g = (\mathbf{R}), \tilde{g} = (\mathbf{R})$ two rotations.

[a] Orthogonal matrices have either determinant $+1$ or -1, special then refers to only considering the $+1$ case.

Parametrization of a group In contrast to the translation case, there is less harm in deciding to directly denote the group elements g with the matrices $\mathbf{R}$ that they represent since the group product and inverse coincide with the matrix product and inverse. However, for notational consistency we usually stick with the group element notation. Moreover, we could decide to group transformations in parametric form, i.e., we could represent 2D rotations with just one number (the rotation angle) instead of a full matrix as follows.

Example 2.3 (2D Rotation group $G = SO(2)$ in parametric form)**.** The 2D rotation group can be represented with the set of rotation angles S^1, i.e., the circle, together with the group product and inverse given by

$$g \cdot \tilde{g} = (\theta + \tilde{\theta}) \,, \tag{17}$$

$$g^{-1} = (-\theta) \,, \tag{18}$$

with $g = (\theta), \tilde{g} = (\tilde{\theta})$. Their matrix representations are given by the rotation matrices

$$\mathbf{R}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} . \tag{19}$$

Note that the groups defined in Example 2.2 with $d = 2$ and the group $SO(2)$ as defined above are equivalent via the relation given in Eq. (19). In fact, the matrix $\mathbf{R}_\theta$ is what one calls a matrix representation of the rotation group $SO(2)$, a concept that we will formalize shortly.

Need for a group structure For both the translation and the rotation group it is not strictly necessary to threat them in a group theoretical setting. As a combination of translation is just obtained by vector addition a vector structure would be sufficient and for rotation we can rely on the algebraic rules for matrix multiplication. However, when considering more general transformations we can no longer treat elements of a transformation group as vectors, which will become apparent in the next example.

Example 2.4 (Rotation-translation group, special Euclidean motion group $SE(d)$)**.** The d-dimensional roto-translation group, also known as the special Euclidean motion group $SE(d)$, consists of the set[a] $\mathbf{R}^d \times SO(d)$ of translation vectors $\mathbb{R}^d$ and rotations $SO(d)$, with group product and inverse given by

$$g \cdot \tilde{g} = (\mathbf{R}\tilde{\mathbf{x}} + \mathbf{x}, \mathbf{R}\tilde{\mathbf{R}}) \,, \tag{20}$$

$$g^{-1} = (\mathbf{R}^{-1}\mathbf{x}, \mathbf{R}) \,, \tag{21}$$

with $g = (\mathbf{x}, \mathbf{R}), \tilde{g} = (\tilde{\mathbf{x}}, \tilde{\mathbf{R}})$.

[a] Here the symbol $\times$ denotes the Cartesian product between sets, i.e., elements of the roto-translation group consist of both a translation vector and a rotation matrix. It should not be confused with the direct product in the group theoretical sense.

We recognize that now we can no longer combine transformation in $SE(2)$ by adding them as vectors or multiplying them as vectors simply because $g = (\mathbf{x}, \mathbf{R})$ is neither just a vector nor a matrix. But even if we were to combine the translations and rotations separately, it wouldn't reflect a proper combination of roto-translations as that would require a mixing of the components of the transformation. In fact, the group $SE(d)$ is what one calls a semi-direct product of the translation group $(\mathbb{R}^d, +)$ with the rotation group $SO(d)$, where semi-direct (formalized in Definition 2.10) refers to the fact that the group elements of the sub-groups interact with eachother in the group product.

Apart from being a semi-direct product group, the above example of $SE(d)$ further justifies the introduction of a group structure as now the combination of two transformations $g, \tilde{g} \in G$ can now simply be written as $g \cdot \tilde{g}$ instead of heaving to write out the product in terms of the computational rules defined for matrices and vectors.

Discrete groups The groups that we consider for now are all Lie groups, however, in order to numerically compute with them we may consider at times discretizations of such groups that form discrete subgroups.

Definition 2.3 (subgroup)**.** A subgroup of a group G is subset $H \subset G$ such that the restriction of the group product from G to H makes H a group.

Example 2.5 (Discrete translation group $G = (\mathbb{Z}^d, +)$)**.** We can think of the set of integer $\mathbb{Z}^d$ vectors as a discretization of $\mathbb{R}^d$, e.g., the pixel grid on which an image may be defined. This set of integers $\mathbb{Z}^d$ together with the addition operator forms a discrete group as the sum of two integers is again an integer. We denote this group with $(\mathbb{Z}^d, +)$. It is a sub-group of $(\mathbb{R}^d, +)$ as $\mathbb{Z}^d \subset \mathbb{R}^d$ and it shares the same group product.

2.2 Group actions and representations (groups acting on vectors and functions)

So far, we have only described how group elements interact with each other through the group product, but not how to interact with other kind of objects such as vectors. This seems a bit limiting, especially when we think of groups as transformations. The group product tells us how to combine transformations, but it doesn't tell us how to apply such transformations to other objects such as vectors or signals. For this, we need a notion of a group action and group representations.

Definition 2.4 (Group action)**.** A group action on a space X is a binary operator $\odot : G \times X \to X$ that follows the group structure (it is a group homomorphism) via

$$g \odot (\tilde{g} \odot x) = (g \cdot \tilde{g}) \odot x\,, \tag{22}$$

with $g, \tilde{g} \in G$ and $x \in X$. Thus, the action can be applied one after the other with two group elements g and $\tilde{g}$ or at once using $g \cdot \tilde{g}$.

Example 2.6 (Group action of rotations $G = SO(d)$ on $\mathcal{X} = \mathbb{R}^d$)**.** The action of the group $G = SO(d)$ (see Example 2.2), denoted with $\odot$, acting on $\mathcal{X} = \mathbb{R}^d$ is given by

$$g \odot \mathbf{x} = \mathbf{R}\mathbf{x}\,, \tag{23}$$

with $g = (\mathbf{R}) \in SO(d)$ and $\mathbf{x} \in \mathbb{R}^d$.

The definition of an action is very general and essentially tells us that a group can act on many types of spaces in many different ways (does not even have to be linearly), as long as the group structure of Eq. (22) is maintained. However, in many cases, specifically in these lecture notes we are only interested in linear group actions that act on vector spaces. Such actions are called representations.

Definition 2.5 (Representation)**.** A representation is an invertible linear transformation $\rho(g) : V \to V$ parametrised by group elements $g \in G$ that acts on some vector space V, and which follows the group structure (it is a group homomorphism) via

$$\rho(g)\rho(h)v = \rho(g \cdot h)v\,,$$

with $v \in V$.

Group acting on $\mathbb{R}^d$ When the vectors in V are expressed in a particular finite dimensional basis the vectors can be represented as elements of $\mathbb{R}^d$ and linear transformations are given by $d \times d$ matrices.

Definition 2.6 (Matrix representation)**.** A matrix representation $\mathbf{D}(g)$ is a representation that acts on a finite dimensional vector space $\mathbb{R}^d$ via $d \times d$ matrices. The matrices are parametrized by group elements and follow the group structure, as required for representations, via

$$\mathbf{D}(g)\mathbf{D}(\tilde{g})\mathbf{x} = \mathbf{D}(g \cdot \tilde{g})\mathbf{x}\,, \tag{24}$$

with $\mathbf{x} \in \mathbb{R}^d$ and $D(g) \in GL(d, \mathbb{R})$.

Group acting on $\mathbb{L}_2(\mathcal{X})$ Recall that the function space $\mathbb{L}_2(X)$ also a vector space (we can add and subtract functions and multiply them with scalars) and thus the action on it is called a representations. The most common way to let a group G act on functions is through left-regular representations. In such representations, a function is transformed simply by transforming the domain $\mathcal{X}$ via the action of the group G on it as follows.

Definition 2.7 (Left-regular representation)**.** Let $f \in \mathbb{L}_2(\mathcal{X})$ and $\odot$ denote the action of the group G on the domain of X. Then the left-regular representation of G acting on $\mathbb{L}_2(\mathcal{X})$ is given by

$$[\mathcal{L}_g f](x) = f(g^{-1} \odot x)\,. \tag{25}$$

It is called left-regular as the group acts on the domain from the left.

Notation of group actions and representations In literature on group equivariant deep learning one regularly encounters an overloading of the group product symbol $\cdot$ to generally represent the action of G on any other object. E.g. denote the action of the group on itself, on $\mathbb{R}^d$, and function space $\mathbb{L}_2(X)$ all with $\cdot$ respectively via $g \cdot \tilde{g}$, $g \cdot \mathbf{x}$, and $(g \cdot f)(\mathbf{x})$ for $g, \tilde{g} \in G$ being group elements, $\mathbf{x} \in \mathbb{R}^d$ and f functions. This makes sense as each of these group actions forms a group themselves[4] that are homomorphic to the group G. Often, however, in it is often helpful to make explicit what kind of objects we are dealing with and so in these lecture we often stick to the following notation

- Action of G on G. The group acts on itself via the group product $\cdot$.
- Action of G on the domain of a function X. We will mostly consider the transformation of signals and their domains. To denote the group action on the domain of such functions we use the $\odot$ notation and e.g. write $g \odot x$ with $x \in X$.
- Action of G on $\mathbb{R}^d$. We may occasionally denote the action of G on $\mathbb{R}^d$ with $\odot$ notation, but mostly we will be explicit and denote the action via matrix representations. E.g., we write $g \odot \mathbf{x} = \mathbf{D}(g)\mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{D}(g) \in GL(d, \mathbb{R})$ a matrix representation of G.
- Action of G on $\mathbb{L}_2(X)$. This is done via the left-regular which we generally denote with $\mathcal{L}_g$.

2.3 Homogeneous space

Homogenous spaces In the introduction Section **??** we approached convolution layers from a template matching point of view, where now with our renewed knowledge of group theory, understand that template matching is nothing else matching a kernel with a signal via the inner product where the kernel is moved around via the group action (or left-regular representation) of G. In order for this construction to be useful, however, it is nice that via the action we can reach every location in the input signal as to be able to exploit all available information. This is guaranteed if the domain $\mathcal{X}$ of the signals $k, f \in \mathbb{L}_2(\mathcal{X})$ is what one calls a homogeneous space of the group G, which we formalize as follows.

Definition 2.8 (Transitive action)**.** A group action $\odot$ of G on a space $\mathcal{X}$ is called transitive if for every pair of points $x, \tilde{x} \in \mathcal{X}$ there exists a group element $g \in G$ such that $g \odot x = \tilde{x}$.

Definition 2.9 (Homogeneous space, semi-direct product, and group quotient)**.** A space $\mathcal{X}$ which has a transitive group action of a Lie group G is called a homogeneous space of G.

[4] You can verify this by checking the group axioms and taking the composition of the action as group product. E.g. the composition of two group actions is again a group actions which verifies closure.

Exercise 2.1. Show that $\mathcal{X} = \mathbb{R}^d$ is a homogeneous space of both $G = (\mathbb{R}^d, +)$ and $G = SE(d)$.

Exercise 2.2. Show that $\mathcal{X} = \mathbb{R}^d$ is *not* a homogeneous space of $G = SO(d)$.

Semidirect product groups In Exercise 2.1 we showed that $\mathbb{R}^d$ is a homogeneous space of $SE(d)$. This is so the case as $\mathbb{R}^d$ can be identified with the translation group $(\mathbb{R}^d, +)$, which is a subgroup of the roto-translationg group $SE(d)$ that is formed as the semi-direct product of translations and rotations.

Definition 2.10 (Semidirect 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

$$(n, h) \cdot (\tilde{n}, \tilde{h}) = (n \cdot (h \odot \tilde{n}), h \cdot \tilde{h}) \,, \tag{26}$$

$$(n, h)^{-1} = (h^{-1} \odot n^{-1}, h^{-1}) \,. \tag{27}$$

Exercise 2.3 ($SE(d) = (\mathbb{R}^d, +) \rtimes SO(d)$)**.** Use the above definition to construct the group $SE(d)$ with group product and inverse

$$g \cdot \tilde{g} = (\mathbf{R}\tilde{\mathbf{x}} + \mathbf{x}, \mathbf{R}\tilde{\mathbf{R}}) \,,$$
$$g^{-1} = (\mathbf{R}^{-1}\mathbf{x}, \mathbf{R}) \,,$$

using the group structure of $(\mathbb{R}^d, +)$ (Example 2.1), the group structure of $SO(d)$ (Example 2.2), and its action on $\mathbb{R}^d$ (Example. 2.6).

Homogeneous spaces of semidirect product groups WRITE SOMETHING HERE In the following we show that

Definition 2.11 (Coset)**.** Let $H \subset G$ be a subgroup of H. Then gH denotes a coset given by

$$gH = \{g \cdot h | h \in H\} \,. \tag{28}$$

Definition 2.12 (Quotient space G/H)**.** Let $H \subset G$ be a subgroup of G. Then G/H denotes the quotient space that is defined as the collection of unique cosets $gH \subset G$. Elements of G/H are thus cosets that represents an equivalence class of transformations for which $g \sim \tilde{g}$ are equivalent if there exsists a $h \in H$ such that $g = \tilde{g}h$.

Lemma 2.1 (G/H is a homogeneous space of G)**.** A quotient space G/H, that has cosets gH as its elements, is a homogeneous space of G. The group G acts transitively on G/H via

$$g \odot \tilde{g}H = (g \cdot \tilde{g})H \,. \tag{29}$$

Exercise 2.4. Show transitivity (Definition 2.8) of the action of G given in Eq. (29).

Example 2.7 (Quotient space $\mathbb{R}^d = SE(d)/SO(d)$)**.** Let $H = (\{\mathbf{0}\} \times SO(d)$ the subgroup of rotations in $SE(d)$, with $\mathbf{0}$ the identity element of the translationg roup $(\mathbb{R}^d, +)$. The the cosets gH are given by

$$\begin{aligned} gH &= \{g \cdot (\mathbf{0}, \tilde{\mathbf{R}}) \mid \tilde{\mathbf{R}} \in SO(d)\} \\ &= \{(\mathbf{R}\mathbf{e} + \mathbf{x}, \mathbf{R}\tilde{\mathbf{R}}) |\ h \in SO(d)\} \\ &= \{(\mathbf{x}, \mathbf{R}\tilde{\mathbf{R}}) |\ \tilde{\mathbf{R}} \in SO(d)\} \\ &= \{(\mathbf{x}, \tilde{\mathbf{R}}) |\ \tilde{\mathbf{R}} \in SO(d)\}\,, \end{aligned}$$

with $g = (\mathbf{x}, \mathbf{R})$. So, the cosets are given by all possible rotations for a fixed translation vector $\mathbf{x}$, the vector $\mathbf{x}$ thus indexes these sets. We can therefore make the identification

$$\mathbb{R}^d \equiv SE(d)/SO(d)\,.$$

We already saw in Exercise 2.1 that $\mathbb{R}^d$ is a homogeneous space of $SE(d)$, this is a consequence of Lemma 2.1.

Lemma 2.1 shows that a quotient space G/H of a group G with H is a homogeneous space. We can also approach this in the other direction and state that any homogeneous space of G is equivalent to a quotient space G/H for some H. This is stated in the following Lemma, for which we first need to introduce the notion of a stabilizer.

Definition 2.13 (Stabilizer)**.** Let G act on X via the action $\odot$. For every $x \in X$, the stabilizer subgroup of G with respect to the point x is denoted with $\mathrm{Stab}_G(x)$ is the set of all elements in G that fix x:

$$\mathrm{Stab}_G(x) = \{g \in G \,|\, g \odot x = x\}. \tag{30}$$

Lemma 2.2. Let X be a homogeneous space of G. Then X can be identified with G/H with $H = \mathrm{Stab}_G(x_0)$ for any $x_0 \in X$.

Affine groups Finally when it comes to types of groups and homogeneous spaces we note that often we are interested in groups that act on $\mathbb{R}^d$, as most often one deal with data on $\mathbb{R}^d$. It is therefore useful to introduce the following class of groups.

Definition 2.14 (Affine groups)**.** Affine groups $G = \mathbb{R}^d \rtimes H$ are a class of groups that are the semidirect product of a group $H \subseteq GL(\mathbb{R}^d)$ acting on $\mathbb{R}^d$, with $GL(\mathbb{R}^d$ the group of general linear transformations acting on $\mathbb{R}^d$.

The transformations in $H \subseteq GL(\mathbb{R}^d)$ are commonly represented as invertible matrices $\mathbf{A}$ which act on $\mathbb{R}^d$ via matrix-vector multiplication, by which the group

product is given by (see ...)

$$g_1 \cdot g_2 = (\mathbf{x}_1, \mathbf{A}_1) \cdot (\mathbf{x}_2, \mathbf{A}_2) = (\mathbf{A}_1\mathbf{x}_2 + \mathbf{x}_1, \mathbf{A}_1\mathbf{A}_2)\,. \tag{31}$$

However, since one often considers H as a subgroup of $GL(\mathbb{R}^d)$ it is typically low dimensional it may be convenient to represent the group elements and product in parametric form.

Example 2.8 (Affine group $SE(2)$)**.** Recall the Example 2.3 for the θ parametrization of $SO(2)$. The group $SE(2)$ is an affine group that consists of the set $\mathbb{R}^2 \times S^1$, where the circle S^1 contains the rotation angles, and which has its group product given by

$$(\mathbf{x}_1, \theta_1) \cdot (\mathbf{x}_2, \theta_2) = (\mathbf{R}_{\theta_1}\mathbf{x}_2 + \mathbf{x}_1, \theta_1 + \theta_2)\,, \tag{32}$$

with $\mathbf{R}_\theta$ the matrix representation of $SO(2)$ (Eq. (19)) acting on $\mathbb{R}^d$.

Affine group convolutions using Conv2D A convenient consequence of working with affine groups (and semidirect product groups in general) is that it allows us to split the group operations into the translation part and the subgroup transformation part. E.g., the representation of such a group $G = \mathbb{R}^d \rtimes H$

$$\mathcal{L}^G_{(\mathbf{x},h)} = \mathcal{L}^{(\mathbb{R}^d,+)}_{\mathbf{x}} \circ \mathcal{L}^G_h\,, \tag{33}$$

where we labeled the representations with the respective groups for which they are defined (G, $(\mathbb{R}^d, +)$, and H). The full group transformation can thus be obtained by first applying the transformation in H, followed by a translation. This allows us to fully leverage the efficiency of the usual (GPU accelerated) $\mathbb{R}^d$ convolution operator as seen in the next example.

Example 2.9 ($SE(2)$ convolutions with $\mathbb{R}^d$ convolutions)**.** Let $k, f \in \mathbb{L}_2(\mathbb{R}^2)$ be a 2D kernel and signal and let us denote correlation on $\mathbb{R}^2$ with $\star_{\mathbb{R}^2}$ as defined in Eq. (10) and the $SE(2)$ lifting correlation with $\star_{SE(2)}$ as defined in Eq. (12). Then the $SE(2)$ lifting correlation is written in terms of $\star_{\mathbb{R}^2}$ correlations as follows

$$\begin{aligned}
(k \star_{SE(2)} f)(\mathbf{x}, \theta) &= (\mathcal{L}^{SE(2)}_{(\mathbf{x},\theta)} k, f)_{\mathbb{L}_2(\mathbb{R}^2)} && (34)\\
&= (\mathcal{L}^{(\mathbb{R}^d,+)}_{\mathbf{x}} \mathcal{L}^{SO(2)}_{\theta} k, f)_{\mathbb{L}_2(\mathbb{R}^2)} && (35)\\
&\overset{*}{=} (\mathcal{L}^{(\mathbb{R}^d,+)}_{\mathbf{x}} k_\theta, f)_{\mathbb{L}_2(\mathbb{R}^2)} && (36)\\
&= (k_\theta \star_{\mathbb{R}^d} f)(\mathbf{x})\,, && (37)
\end{aligned}$$

where in step $\overset{*}{=}$ we defined $k_\theta(\mathbf{x}) = k(\mathbf{R}_\theta^{-1}\mathbf{x})$ as the rotated kernel. So, we see that a lifting group convolution can be obtained by first precomputing a filter bank of rotated kernels and apply to the input signal via the usual correlation operator, which in many deep learning libraries is efficiently implemented in a Conv2D layer.

2.4 Equivariance

Now we have set all the prerequisites in place to formalize the term equivariance. Equivariance is a property of an operator (for example a layer in a neural network) that guarantees that if the input transforms the output transforms in a predictive way.

Definition 2.15 (Equivariance)**.** An operator $\Phi : \mathcal{X} \to \mathcal{Y}$ that sends elements from input space $\mathcal{X}$ to output space $\mathcal{Y}$ is called equivariant to a group G if there are actions of G on $\mathcal{X}$ and $\mathcal{Y}$ respectively denoted by $\rho^{\mathcal{X}}$ and $\rho^{\mathcal{Y}}$ such that

$$\forall_{g \in G}: \quad \rho^{\mathcal{Y}}(g) \circ \Phi = \Phi \circ \rho^{\mathcal{X}}(g). \tag{38}$$

In other words, the operator Φ commutes with actions of the group G.

In a deep learning context this is a very useful, if not essential, property to have. For example, if a layer is translation equivariant, such as the convolution/correlation layer, then this means that if the input were to be translated that the output is translated as well. This implies the following:

1. Equivariance implies that no information is lost by input transformations, it is just shifted to different locations in the output.

2. Equivariance allows for weight sharing over the transformations in the group. Being able to detect a feature at one location/pose guarantees that it is equally well detected at another location.

Example 2.10 (Convolution layer)**.** To link Definition 2.15 more precisely to the CNN setting consider the following. Let a layer Φ_k be

$$f^{out}(\mathbf{x}) = [\Phi_k f^{in}](\mathbf{x}) = (k \star_{\mathbb{R}^d} f^{in})(\mathbf{x}),$$

defined by a correlation operator as defined in Eq. (10) and parametrized by kernel k. Then Φ_k is equivariant to the translation group $G = (\mathbb{R}^d, +)$ as it maps input feature maps $f^{in} \in \mathcal{X} = \mathbb{L}_2(\mathbb{R}^d)$ to output feature maps $f^{out} \in \mathcal{Y} = \mathbb{L}_2(\mathbb{R}^d)$ on which the representation of the translation group acts via $\rho^{\mathcal{X}}(g) = \rho^{\mathcal{Y}}(g) = T_g$ as given in Eq. (9).

Exercise 2.5. Proof that the correlation operator is equivariant to the translation group, as claimed in Example 2.10.

2.5 Integration on G using the Haar measure

Finally, in order to have all the mathematical tools in place to define the general framework for regular G-CNNs in Chapter 3 we need to know how to integrate over groups, as required in convolutions, inner products, or norms. Therefore, we need a measure defined on G.

Definition 2.16 (Haar measure, unimodular groups)**.** A Haar measure $\mathrm{d}g$ is an invariant measure on the group G. The following defines two types of Haar measures.

- The *left Haar measure* is invariant to left multiplications, i.e.,

$$\forall_{\tilde{g}\in G}:\ \mathrm{d}(\tilde{g}g) = \mathrm{d}g.$$

- The *right Haar measure* is invariant to right multiplications, i.e.,

$$\forall_{\tilde{g}\in G}:\ \mathrm{d}(g\tilde{g}) = \mathrm{d}g.$$

Groups for which the left and right Haar measure coincide are called unimodular groups. The Haar measure on unimodular groups is both left- and right-invariant.

The Haar measure plays a central role in proving equivariance of differential operators as it allows the use integration by substitution whilst leaving the measure invariant. For example we can use it to derive the following useful identity

Lemma 2.3. Let $k, f \in \mathbb{L}_2(G)$ and $\mathcal{L}_g$ the left regular representation of a group G on $\mathbb{L}_2(G)$, with left Haar measure dg. Then we have

$$(\mathcal{L}_g k, f)_{\mathbb{L}_2(G)} = (k, \mathcal{L}_{g^{-1}} f)_{\mathbb{L}_2(G)}\,.$$

Proof. The proof is obtained via as follows:

$$\begin{aligned}(\mathcal{L}_g k, f)_{\mathbb{L}_2(G)} &= \int_G [\mathcal{L}_g k](\tilde{g}) f(\tilde{g}) \mathrm{d}\tilde{g} = \int_G k(g^{-1}\tilde{g}) f(\tilde{g}) \mathrm{d}\tilde{g} \\ &\overset{*}{=} \int_{gG} k(\tilde{g})(g\tilde{g}) \mathrm{d}g\tilde{g} = \int_G k(\tilde{g})[\mathcal{L}_{g^{-1}} f](\tilde{g}) \mathrm{d}\tilde{g} \\ &= (k, \mathcal{L}_{g^{-1}} f)_{\mathbb{L}_2(G)}\,,\end{aligned}$$

where $\overset{*}{=}$ the substitution $\tilde{g} \mapsto g\tilde{g}$ is made. □

Exercise 2.6. Let $k, f \in \mathbb{L}_2(\mathbb{R}^d)$ and $G = SE(d)$. The group action of $SE(d)$ on $\mathbb{R}^d$ given by

$$g \odot \tilde{\mathbf{x}} = \mathbf{R}\tilde{\mathbf{x}} + \mathbf{x}\,,$$

with $g \in SE(d)$ and $\mathbf{x} \in \mathbb{R}^d$. Then let $\mathcal{L}_g$ be the left-regular representation of $SE(d)$ on $\mathbb{L}_2(\mathbb{R}^d)$. Proof the following:

$$(\mathcal{L}_g k, f)_{\mathbb{L}_2(\mathbb{R}^d)} = (k, \mathcal{L}_{g^{-1}} f)_{\mathbb{L}_2(\mathbb{R}^d)}. \tag{39}$$

Exercise 2.7. Consider Φ_k the roto-translation lifting correlation parametrized by kernel $k \in \mathbb{L}_2(\mathbb{R}^2)$ as defined in Eq. (12) and the definition of equivariance as in Definition 2.15.

(a) What are the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ of Φ_k?

(b) Give the representations $\rho^{\mathcal{X}}$ and $\rho^{\mathcal{Y}}$ of $SE(2)$ that acts on $\mathcal{X}$ and $\mathcal{Y}$.

(c) Proof that Φ_k is equivariant to the roto-translation group $SE(2)$ using the identity given in Eq. (39).

Lemma 2.4 (Duality relation for affine groups acting on $\mathbb{L}_2(\mathbb{R}^d)$)**.** For affine groups $G = \mathbb{R}^d \rtimes H$ with group elements $g = (\mathbf{x}, h) \in G$ the following identity holds

$$(\mathcal{L}_g k, f)_{\mathbb{L}_2(\mathbb{R}^d)} = |\det h| \, (k, \mathcal{L}_{g^{-1}} f)_{\mathbb{L}_2(\mathbb{R}^d)} \,, \tag{40}$$

with $|\det h|$ the absolute determinant (Jacobian) of the matrix representation of h acting on $\mathbb{R}^d$.

Exercise 2.8. Proof Lemma 2.4.

The $|\det h|$ factor is a result of the integration by substitution in which one has to take the Jacobian of the transformation into account. The fact that the identity of Eq. (39) holds for $SE(2)$ is that it is a unimodular group for which $|\det h| = 1$. Note, however, that when integrating over the group G, e.g., when considering the $\mathbb{L}_2(G)$ inner product, we do not have such a front factor as there we integrate using the invariant Haar measure and have

$$(\mathcal{L}_g k, f)_{\mathbb{L}_2(G} = (k, \mathcal{L}_{g^{-1}} f)_{\mathbb{L}_2(G)} \,. \tag{41}$$

Haar measure and parametrizations One can make the Haar measure explicit in terms of the parametrization used for the group. Will include next version of the notes.

3 Regular Group Convolutional Neural Networks

Before we move on to the details let us start by stating the most important message of these lecture notes.

A linear layer between feature maps
is equivariant if and only if it
is a group convolutions.

Recall that the success of deep learning is largely due to the introduction of convolutional neural networks (CNNs). The impact of CNNs can be attributed to the equivariance property the convolution layers by which they are build. Namely,

1. Equivariance implies that no information is lost by input transformations, it is just shifted to different locations in the output.

2. Equivariance implies that we can guarantee that data points that equivalent up to a transformation are treated in the same way. E.g. we still say that an image contains a cat regardless of whether the cat is in the upper left, or bottom right corner of the image.

3. Equivariance allows for weight sharing over the transformations in the group. Being able to detect a feature at one location/pose guarantees that it is equally well detected at another location. This reduces the number of learnable parameters as now one does not have to relearn what a feature looks like at each location.

4. Equivariant operators preserve geometric structure in the data. Structure may refer to the neat ordering of pixels an image that allow us to identify neighborhoods and have a sense of locality that allows for weight sharing. The fact that equivariant operators preserve the geometric structure of the data allow us to build *deep* equivariant architectures.

In many cases there is more structure than just translational symmetries. It is the purpose of these lecture notes to provide methods for extending the equivariance property beyond just translations by explaining the framework of group equivariant CNNs, or simply G-CNNs.

As an example for the need for equivariance beyond translations consider the following. In medical image data there often is no intrinsic/preferred orientation and tasks such as tumor or blood vessel detection are inherently roto-translation invariant; one wants to be able to detect tumors or blood vessels regardless of how they are oriented in the data. The same is the case in physical systems or in computational chemistry. E.g., in N-body problems where particles interact

with each other in an equivariant manner. When the system of particles rotate, the forces rotate accordingly.

There are too many applications to list here where equivariance plays a key role. The main message here is, it only requires a few moments of thought to figure out which geometric structure underlies your data and problem and realize that you may want to preserve it for the reasons mentioned above. It then follows that the best you can do is work with group convolutional neural networks since, as we will show shortly: The group convolution operator is the most general/expressive linear operator that is equivariant.

3.1 Artifical Neural Networks for Vectors

Let us first consider the classical structure of artificial neural networks (NNs). Classically NNs are build for the processing of vectors in $\mathbb{R}^{N_c}$, where these vectors are iteratively transformed by interleaving linear (affine) transformations with non-linear activation functions. Let l index the depth of the neural network, and $\mathbf{x}^l \in \mathbb{R}^{N_l}$ the feature vectors at layer l. Then a layer Φ^l is typically defined as

$$\mathbf{x}^{l+1} = \Phi^l(\mathbf{x}^l) = \sigma(\mathbf{W}^l\mathbf{x}^l + \mathbf{b}^l)\,, \tag{42}$$

in which $\mathbf{W}^l \in \mathbb{R}^{N_{l+1}\times N_l}$ is a dense matrix with learnable weights, $\mathbf{b}^l \in \mathbb{R}^{N_l}$ a leanable bias vector, and σ a non-linear activation function that is applied element wise.

It should be clear that processing images, or signals in general, via the defined layers such as the above the geometric structure of the data is completely lost as there is no guarantee/constraint that neighboring pixels are processed in a similar way. It turns out that if we do impose an equivariance constraint on Eq. (42) we obtain a discrete implementation of a (group) convolution.

Example 3.1 (Equivariant layer for discrete periodic signal)**.** Consider a periodic signal $f \in \mathbb{L}_2(S^1)$ on S^1 that is sampled on a discrete grid $\mathbf{X} = \{0, 2\pi/N, \dots, 2\pi - 2\pi/N\}$ with N points. The signal is then represented with a vector $\mathbf{f}^0 = (f(x_1), f(x_2), \dots, f(x_N))^T \in \mathbb{R}^N$. Now lets denote a periodic shift with the permutation matrix

$$\mathbf{S} = \begin{pmatrix} 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & 0 & 0 & \dots \end{pmatrix}.$$

Then, wanting $\Phi(\mathbf{S}\mathbf{x}^l) = \mathbf{S}\Phi(\mathbf{x}^l)$ implies that $\mathbf{W}$ should commute with $\mathbf{S}$, i.e., $\mathbf{W}\mathbf{S} = \mathbf{S}\mathbf{W}$. Solutions to this problem are given by circulant matrices of the form

$$\mathbf{W} = \begin{pmatrix} w_1 & w_2 & w_3 & \dots \\ w_N & w_1 & w_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ w_2 & w_3 & w_4 & \dots \end{pmatrix},$$

in which each row is a shifted weight vector $\mathbf{w} \in \mathbb{R}^N$ that is shared over all rows. The implications are threefold:

- Equivariance is guaranteed meaning that the same features are represented regardless of the initial pose/shift of the input. Upon a shift of the signal the output is just shifted in a predictable way.
- The number of learnable parameters is drastically reduced as now the layer is parametrized by N weights instead of N^2.
- Since the (neighborhood) structure is preserved on can choose to localize the transformation. I.e., by e.g. setting $w_n = 0$ for $n > 3$ one ensures that the output at each location is only generated using a neighborhood of 3 input points. Which give an even further reduction of parameters. Moreover, on non-periodic signals this reduces boundary artifacts.

3.2 Artificial Neural Networks for Signals

Linear transformations of signals In a similar way as we build NNs to process vectors by interleaving linear layers with non-linearities. We can make a construction for NNs to process continuous signals. Like in the case of classical NNs, the main workhorse will be the linear layer, that now has to transform feature maps $f \in \mathbb{L}_2(X)$ instead of vectors $\mathbf{f} \in \mathbb{R}^N$. Whereas for vector spaces the most general linear transormations are given by matrix vector multiplications, for continuous signals linear transformations are given by kernel transformations.

Theorem 3.1 (Dunford-Pettis: Linear bounded maps are integral transforms)**.** Let $\mathcal{K} : \mathbb{L}_2(X) \to \mathbb{L}_2(Y)$ be linear and bounded operator that maps between spaces of feature maps $\mathbb{L}_2(X)$ and $\mathbb{L}_2(Y)$. Then there exists a kernel $k \in \mathbb{L}_1(Y, X)$ such that $\mathcal{K}$ is an integral transform via

$$(\mathcal{K}f)(y) = \int_X k(y,x) f(x) \mathrm{d}\mu_X(x)\,, \tag{43}$$

with $f \in \mathbb{L}_2(X)$, and $\mathrm{d}\mu_X$ a Radon measure on X.

Intuitively one may think of Eq. (43) as the continuous counter part of matrix vector multiplication, having in mind that signals are also vectors, be it infinite-dimensional. Like in matrix-vector multiplication the output vector at index j is given by a sum over the product of the matrix coefficients with all elements of the input vector indexed with i. In the continuous counter part, the vectors (signals) are indexed with continuous "indices" x and y, the summation is an integral, and the matrix is a two argument kernel:

$$\text{Vectors:} \qquad \mathbf{f}^{l+1} = \mathbf{W}\mathbf{x} \qquad \Leftrightarrow \qquad f^{l+1}_j = \sum_i^{N_l} w_{ji}\, x^l_i\,, \tag{44}$$

$$\text{Signals:} \qquad f^{l+1} = \mathcal{K} f^l \qquad \Leftrightarrow \qquad f^l(y) = \int_X k(y,x) f^l(x) \mathrm{d}\mu_X(x)\,. \tag{45}$$

Linear layers for multi-channel signals An artificial neural network for processing feature maps would than be based on layers that map one multi-channel feature map $\underline{f}^l \in \mathbb{L}_2(X^l)^{N_l}$ to the next one $\underline{f}^{l+1} \in \mathbb{L}_2(X^{l+1})^{N_{l+1}}$ via a layers of the following form

$$\underline{f}^{l+1} = \sigma(\boldsymbol{\mathcal{K}}\underline{f}^l + \mathbf{b}^l)\,, \tag{46}$$

with learnable bias vector $\mathbf{b} \in \mathbb{R}^{N_{l+1}}$, and in which $\boldsymbol{\mathcal{K}}$ denotes a kernel operator as defined for multi-channel signals via

$$\boldsymbol{\mathcal{K}}\underline{f}^l = \int_X \mathbf{K}(y,x)\underline{f}^l(x) \mathrm{d}\mu_X(x)\,, \tag{47}$$

in which the layer is parametrized by a learnable matrix-valued kernel $\mathbf{K} : Y \times X \to \mathbb{R}^{N_{l+1} \times N_l}$ that for each y, x combination transforms the feature vectors $\underline{f}^l(x) \in \mathbb{R}^{N_l}$. In the above we indexed the domain of the feature maps with l, which may seem odd as it suggests that one may want to change it as a function of depth. We consider this option since it makes sense in the group equivariant framework where we may want to lift signals on some domain, e.g., $X^l = \mathbb{R}^d$ to a Lie group, e.g., $X^{l+1} = SE(2)$. Irrespective of group equivariance, one can think of a change of the domain of the signals also from a discretization perspective where a coarsening of the grid with depth l corresponds to downsampling/strides.

3.3 Equivariant Neural Networks for Signals

As motivated before, we want to build equivariant neural networks for continuous signal data. This means that the layers that we will be using, such as those of Eq. 46, to be equivariant. With the following theorem we show that our only option then will be to use group convolutions. In order not to clutter notation we will omit the symbols $\cdot$ and $\odot$ as we have been doing so far, but simply write $g\tilde{g}$ and gx to denote the product and action respectively.

Theorem 3.2 (Equivariant linear layers on homogeneous spaces)**.** Let operator $\mathcal{K} : \mathbb{L}_2(X) \to \mathbb{L}_2(Y)$ be linear and bounded, let X, Y be homogeneous spaces on which Lie group G act transitively, and $\mathrm{d}\mu_X$ a Radon measure on X. Let furthermore $\mathcal{K}$ be constrained to be equivariant to the group G via $\mathcal{L}^Y_g \circ \mathcal{K} = \mathcal{K} \circ \mathcal{L}^X_g$, with $\mathcal{L}^X_g$ and $\mathcal{L}^Y_g$ the left-regular representations of G on respectively $\mathbb{L}_2(X)$ and $\mathbb{L}_2(Y)$.

1. Then $\mathcal{K}$ is a group convolution given by

$$(\mathcal{K}f)(y) = \int_X \frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} k(g_y^{-1}x) f(x) \mathrm{d}\mu_X(x), \tag{48}$$

   for any $g_y \in G$ such that $y = g_y y_0$ for some fixed origin $y_0 \in Y$.

2. The kernel has to satisfy the following symmetry constraint

$$k(x) = \frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} k(h^{-1}x), \tag{49}$$

   for any $h \in H = \mathrm{Stab}_G(y_0)$ and any $x \in X$.

Proof. See [Bekkers, 2019, App. A] for a complete derivation. A sketch of the poof is as follows. Theorem 3.1 tells us that $\mathcal{K}$ is an integral transform. The equivariance constraint then imposes that the two-argument kernel of $\mathcal{K}$ is left-invariant via

$$\forall_{g \in G} : \quad k(y, x) = \frac{\mathrm{d}\mu_X(g^{-1}x)}{\mathrm{d}\mu_X(x)} k(g^{-1}y, g^{-1}x)\,. \tag{50}$$

Transitivity of the group G on Y implies that we can define a $g_y \in G$ such that $y = g_y y_0$ by which we can use Eq. (50) to turn the two-argument kernel effectively a one-argument kernel via $k(g_y^{-1}x) := k(y_0, g_y^{-1}x)$ for some fixed $y_0 \in Y$. This proofs the first item.

The kernel constraint follows from the fact that every homogeneous space Y of G can be identified with a quotient group G/H. Choose an origin $y_0 \in Y$ s.t. $\forall_{h \in H} : h\, y_0 = y_0$, i.e., $H = \mathrm{Stab}_G\, y_0$, then

$$\forall h \in H : \quad k(y_0, x) = k(h\, y_0, x) \Leftrightarrow k(x) = \frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} k(h^{-1}\, x)\,.$$

□

In Theorem 3.2 we grayed out the front factor $\frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)}$ as in many cases it greatly simplifies, or can even be completely omitted.

Corollary 3.2.1. The following lists special cases in which the front factor $\frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)}$ is greatly simplified:

- $\frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} = 1$ for unimodular groups G.
- $\frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} = 1$ when $X = G$ the group itself and using the left Haar measure for $\mathrm{d}\mu_X(g) = \mathrm{d}g$.
- $\frac{\mathrm{d}\mu_X(g_y^{-1}x)}{\mathrm{d}\mu_X(x)} = \frac{1}{|\det h|}$ when $G = \mathbb{R}^d \rtimes H$ is an affine group and when $\mathrm{d}\mu_X(x)$ is the Lebesgue measure $\mathrm{d}x$. Here $\det h$ denotes the determinant of the matrix representation of H on $\mathbb{R}^d$.

Exercise 3.1 (Kernel constraint)**.** Derive the left-invariance constraint of the two-argument kernel as given in Eq. (50) using integration by substitution of variables.

3.4 Types of equivariant layers

In Theorem we have several options to choose for the homogeneous spaces X, the domain on which the input signal lives, and Y, the domain on which we want to output signal to live. Let us make these options explicit for affine groups $G = \rtimes\mathbb{R}^d \rtimes H$ with the following definitions.

3.4.1 Isotropic $\mathbb{R}^d$ convolutions ($X = Y = \mathbb{R}^d$)

When one sticks to working with planar feature maps, as is common in standard CNNs that are build with operators such as Conv1D, Conv2D, etc., then the correlations are planar correlations as usual but the kernels are constraint to be invariant to transformations of H. E.g., when considering $G = SE(d)$, for which $H = SO(d)$, one comes to the well-known result that only isotropic kernels can be used in order to guarantee rotation equivariance. Note, however, that it is not always possible to construct H-invariant kernels such as e.g. is the case with the dilation/scaling group $H = (\mathbb{R}^+, \cdot)$ where there are no solutions to Eq. (49).

Definition 3.1 (Isotropic $\mathbb{R}^d$ convolutions ($X = Y = \mathbb{R}^d$))**.** An isotropic $\mathbb{R}^d$ convolution layer maps between planar signals $\mathbb{L}_2(\mathbb{R}^d)$ by layers given by Eq. (46) with $\mathcal{K}$ a planar correlation given by

$$(\mathcal{K}f)(\mathbf{y}) = \int_{\mathbb{R}^d} \tfrac{1}{|\det h|} k(\mathbf{x}-\mathbf{y}) f(\mathbf{x}) \mathrm{d}\mathbf{x}\,, \tag{51}$$

and in which k satisfies

$$\forall h \in H: \quad k(x) = \tfrac{1}{|\det h|} k(h^{-1}x)\,.$$

3.4.2 Lifting layer ($X = \mathbb{R}^d$, $Y = G$)

Isotropic convolutions are rather limiting due to the kernel constraint. It is therefore sensible to lift the signals to the group G, as then $H = \emptyset$ and there are not restrictions on k. Consequently, the kernel needs to be matched with the input signal for every possible transformation in G. E.g. in the $SE(2)$ case, we want to rotate the kernel for every possible rotation as not to miss out the detection of a feature at a particular orientation. By making sure the kernel is matched for all possible transformation we guarantee that the transform maps are equivariant, namely all relevant information will be encoded in the output feature map, it is just that under a transformation of G on the input the information may end up at a different location in the output. We will refer to such lifted feature maps (functions on G) as G-feature maps.

Definition 3.2 (Lifting layer ($X = \mathbb{R}^d$, $Y = G$))**.** A lifting layer maps from planar signals $\mathbb{L}_2(\mathbb{R}^d)$ to signals $\mathbb{L}_2(G)$ on the group G. It follows the general form of the layer as given by Eq. (46), with $\mathcal{K}$ a lifting correlation given by

$$(\mathcal{K}f)(g) = \int_{\mathbb{R}^d} \tfrac{1}{|\det h|} k(g^{-1}\tilde{\mathbf{x}}) f(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}\,. \tag{52}$$

A note on implementation Recall from Eq. (33) and Example 2.9 that such lifting correlations can be implemented by ConvdD operators with a bank of H-transformed kernels $k_h(\mathbf{x}) = k(h^{-1}\mathbf{x})$. I.e.,

$$(\mathcal{K}f)(\mathbf{x}, h) = (k_h \star_{\mathbb{R}^d} f)(\mathbf{x})\,.$$

3.4.3 Group convolution layer

($X = G$, $Y = G$) The output of the lifting layer is a function on the group G. If we then wish to proceed with layers without any constraints on k we again map them to G-feature maps ($Y = G$). The principle remains the same as always, do template matching with a kernel over all transformations in G. Now, however, the input is a higher-dimensional signal and so will k be in $\mathbb{L}_2(\mathbb{R}^d \times H)$.

Definition 3.3 (Group convolution layer ($X = Y = G$))**.** A group convolution layer maps between G-feature maps in $\mathbb{L}_2(G)$. It follows the general form of the layer as given by Eq. (46), with $\mathcal{K}$ a group correlation given by

$$(\mathcal{K}f)(g) = \int_G k(g^{-1}\tilde{g}) f(\tilde{g}) \mathrm{d}\tilde{g}\,, \tag{53}$$

with $\mathrm{d}\tilde{g}$ the left Haar measure on G.

A note on implementation Note that also these layers can be implemented using ConvdD operators by making a bank of transformed convolution kernels,

and by merging the discretized H axis with the channel axis as to move integration over H into the summation over the channels in the ConvdD operator.

Consider the $G = SE(2)$ case. Now the feature maps are three-dimensional and assign to every possible translation and rotation a feature value. The kernels are then also three-dimensional $k : \mathbb{R}^2 \times S^1 \to \mathbb{R}^{N_{l+1} \times N_l}$ and assign to every relative position and orientation a feature transformation. In numerical implementations we discretize the domain X and the integral becomes a summation. If we decouple the convolution in a spatial and a rotation part it is given for $SE(2)$ as

$$(\mathcal{K}f)(\mathbf{x}, \theta) = \int_{\mathbb{R}^2} \int_{S^1} \mathbf{k}(\mathbf{R}_\theta^{-1}(\tilde{\mathbf{x}} - \mathbf{x}), \tilde{\theta} - \theta) \underline{f}(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \tilde{\theta} \,. \tag{54}$$

In a numerical discretization with N_h elements sampled on the H axis, this would be implemented with Conv2d with a bank of transformed filters, denoted with $\mathbf{K} : \mathbb{R}^2 \to \mathbb{R}^{(NN_{l+1}) \times (NN_l)}$, and of which also the the H and feature axis of the input are merged. I.e., usually the feature map $\underline{f}$ would be stored as a tensor in $\mathbb{R}^{N_x \times N_y \times N_h \times N_l}$ that represents a function $\mathbb{R}^2 \times H \to N_l$.

3.4.4 Projection layer ($X = G$, $Y = \mathbb{R}^d$)

By mapping backing to planar feature maps $\mathbb{L}_2(\mathbb{R}^d)$ one obtains locally H-invariant feature maps given as follows.

Definition 3.4 (Projection layer ($X = G$, $Y = \mathbb{R}^d$))**.** A projection layer maps between G-feature maps in $\mathbb{L}_2(G)$ back to planar feature maps in $\mathbb{L}_2(\mathbb{R}^d)$. It follows the general form of the layer as given by Eq. (46), with $\mathcal{K}$ a pooling over H given by[a]

$$(\mathcal{K}f)(g) = \int_H f(\mathbf{x}, \tilde{g}) \mathrm{d}\tilde{h} \,. \tag{55}$$

[a] Actually it may still include an isotropic kernel that is constant over the h axis. I will make this precise in a future version of the notes.

3.4.5 Global pooling ($X = G$, $Y = \emptyset$)

Often one is interested in building invariant architectures, rather than equivariant. Invariance to all transformations in G is achieved by mean pooling over the full group G, in a similar way as one usually achieves global translation invariance by mean or max pooling over the spatial axes in feature maps. Namely, in this case $H = G$ would be the group itself and thus the kernel constraint tells us that the kernel is constant over G, i.e., $\forall_{g \in G} k(e) = k(g) = c$ for some $c \in \mathbb{R}$. Instead of using this trival k in the equation we may simply define gobal pooling as follows.

Definition 3.5 (Global pooling layer ($Y = \emptyset$))**.** A gobal pooling layer maps any feature map in $\mathbb{L}_2(X)$ back to a single scalar value. It follows the general form of the layer as given by Eq. (46), with $\mathcal{K}$ a pooling over X

$$(\mathcal{K}f) = \int_X f(x)\mathrm{d}\mu_X(x)\,. \tag{56}$$

3.5 Designing equivariant neural networks

Networks build with isotropic $\mathbb{R}^d$ layers follow the exact same structure as standard CNNs and exactly the same in terms of computational complexity. However, these isotropic CNNs are guaranteed to be equivariant roto-translations whereas the classical ones only to translations. This comes at the cost of expressiveness as the kernels are constraint to be isotropic.

The most expressive networks are build with group equivariant layers which have no constraints imposed on them (i.e. $Y = G$). Such networks are referred to as group convolutional neural networks as the group convolution layer (Definition 3.3) dominates those networks. No constraints means $Y = G$, i.e., lifting the data to G-feature maps instead of the standard $\mathbb{R}^d$-feature maps. The general structure for problems like segmentation and classification is then

$$\begin{aligned}\mathbb{R}^d\text{-feature map} &\overset{\text{lifting layer}}{\rightarrow} G\text{-feature map} \overset{\text{repeated g-conv layers}}{\rightarrow} \\ &G\text{-feature map} \overset{\text{projection layer}}{\rightarrow} \mathbb{R}^2\text{-feature map (segmentation)}\,,\end{aligned}$$

and

$$\begin{aligned}\mathbb{R}^d\text{-feature map} &\overset{\text{lifting layer}}{\rightarrow} G\text{-feature map} \overset{\text{repeated g-conv layers}}{\rightarrow} \\ &G\text{-feature map} \overset{\text{global } G\text{-pooling}}{\rightarrow} \text{scalar number (classification)}\,.\end{aligned}$$

It is important to remark that, even when the problem at hand does not require full group equivariance/invariance, group equivariant layers are still beneficial. This is due to the weightsharing property over groups G larger than translations, and the fact that the higher dimensional kernels (function over G instead of $\mathbb{R}^d$) can express more complicated patterns. E.g., computer vision tasks are often not truely rotation invariant as there is often a horizon and a direction of gravity that aligns objects (people typically appear upright in images). Nevertheless, the core features (edges, corners, ...) that make up the more high level objects can appear under all possible orientations or scales in images. The higher level G-conv kernels then can describe advanced patterns in terms of configurations of features at both positions/translations and orientations/rotations relative to each other. This solves the "Picasso problem" in an equivariant manner: when detecting a face, the eyes and nose all have to be at the right place *and* right orientation, and not just the right place.

Part II
Steerable group convolutional neural networks

In this part we will cover a particular class of roto-translation equivariant group convolutions called *steerable group convolutions*. Steerable group convolutions still fall in the class of general equivariant operators as presented in Chapter 3, they are linear and equivariant and thus group convolutions (Theorem 3.2). What makes them special, however, is the fact that they enable implementations of group convolutions without having to discretize the sub-group of rotations $H = SO(d)$.

The property that discretizations of $SO(d)$ can now be avoided is particularly useful in the three-dimensional case. Whereas for $d = 2$ we can discretize the rotation group with arbitrary precision, and it will create a discrete subset of points which is again a (sub)group. We can then guarantee[5] equivariance to the subgroup. For $SO(3)$ we do not have the luxery that any discretization of $SO(2)$ (as a set) is again a subgroup. We could e.g. make a grid of N arbitrary elements, but this would not form a discrete group (we do not necessarily have $g \cdot \tilde{g} \in \mathbf{G}$ for $g, \tilde{g} \in \mathbf{G}$ with $\mathbf{G}$ discretization of G.). We would then have to resort to interpolation on $SO(3)$ which can be expensive and it introduces numerical errors. Only a few special cases $N = 12, 24$, or 60) for which we can make a discrete subgroup. E.g., we could discretize $SO(3)$ with the symmetries of a cube ($N = 24$). This is a closed subgroup and it would guarantee equivariance as no interpolation would have to be done. It would however, only be equivariant to this subgroup and not the full $SO(3)$.

Steerable methods avoid a discretization of $SO(d)$ by means of irreducible representations, a type of representations that allows for Fourier theory on Lie groups. We will introduce the notion of irreducible representations shortly. We begin with Section 4 introducing a new set of group theoretical definitions that allow us to formalize the notion of steerability. As done before in Part I, we will accompany the definitions with examples and follow it with a section dedicated to discussion of the theory in a deep learning context in Section 5.

[5] Up to spatial interpolation artifacts.

4 Mathematical tools: representation theory for $SO(3)$ and the Clebsch-Gordan tensor product

In this section we introduce the mathematical tools and intuition for steerable methods. It is heavily influenced by works such as [Thomas et al., 2018, Anderson et al., 2019, Fuchs et al., 2020], which the reader may appreciate as excellent alternative resources[6] to get acquainted with the group/representation theory used in steerable methods. We focus on the following main conceptsThe main concepts

1. *Steerable vectors*, *Wigner-D matrices* and *irreducible representations* (Section 4.2). Whereas regular NNs work with feature vectors whose elements are scalars, our steerable NNs work with feature vectors consisting of steerable feature vectors. Steerable feature vectors are vectors that transform via so-called Wigner-D matrices, which are representations of the orthogonal group O(3). Wigner-D matrices are the smallest possible group representations and can be used to define any representation (or conversely, any representation can be reduced to a tensor product of Wigner-D matrices via a change of basis). As such, the Wigner-D matrices are irreducible representations.

2. *Spherical harmonics* (Section 4.3). Spherical harmonics are a class of functions on the sphere S^2 and can be thought of as a Fourier basis on the sphere. We show that spherical harmonics are steered by the Wigner-D matrices and interpret steerable vectors as functions on S^2. Moreover, spherical harmonics allow the embedding of three-dimensional displacement vectors into arbitrarily large steerable vectors.

3. *Clebsch-Gordan tensor product* and *steerable linear layers* (Section 4.4). In a regular NN one maps between input and output vector spaces linearly via matrix vector multiplication and applies non-linearities afterwards. In steerable NNs one maps between steerable input and steerable output vector spaces via the Clebsch-Gordan tensor product. Akin to the learnable weight matrix in regular NNs, the learnable Glebsch-Gordan tensor product is the workhorse of our steerable NNs.

After these concepts are introduced we will in Section 5 revisit the convolution operator in the light of the steerable, group theoretical viewpoint that we take in this paper. In particular, we show that steerable group convolutions are equivalent to linear group convolutions with convolution kernels expressed in a spherical harmonic basis.

[6] Each of these works presents unique view points that greatly influenced the writing of this appendix.

4.1 The groups $E(3)$ and $O(3)$ and homogeneous space S^2

4.1.1 The groups $E(3)$ and $O(3)$

The Euclidean group E(3) In the steerable setting, we are interested in the group of three-dimensional translations, rotations, and reflections which is denoted with $E(3)$, the 3D Euclidean group. It is an affine group $E(3) = (\mathbb{R}^d, +) \rtimes O(3)$ that is slightly larger than the previuosly discussed group $SE(3)$; the subgroup $O(3)$ also contains reflections ($\det \mathbf{R} = -1$. The group structure (product, action on $\mathbb{R}^d$ is otherwise the same.

The orthogonal group O(3) and special orthogonal group SO(3). The group $\mathrm{E}(3) = \mathbb{R}^3 \rtimes \mathrm{O}(3)$ is a semi-direct product of the group of translations $\mathbb{R}^3$ with the group of orthogonal transformations $O(3)$. This means that we can conveniently decompose $E(3)$-transformations in an $O(3)$-transformation (rotation and/or reflection) followed by a translation. As such we can focus on the more complicated structure of $O(3)$, as the translation part can be dealt with using the standard machinery (e.g. the use of Conv3D).

4.1.2 The sphere S^2 is a homogeneous space of $O(3)$.

The sphere is not a group as we cannot define a group product on S^2 that satisfies the group axioms. It can be convenient to treat it as a homogeneous space of the groups $O(3)$ or $SO(3)$. The sphere S^2 is a homogeneous space of the rotation group $O(3)$ since any point on the sphere can be reached via the rotation and/or reflection of some reference vector.

Consider for example an XYX parametrisation of $SO(3)$ rotations in which three rotations are applied one after another via

$$\mathbf{R}_{\alpha,\beta,\gamma} = \mathbf{R}_{\alpha,\mathbf{n}_x}\mathbf{R}_{\beta,\mathbf{n}_y}\mathbf{R}_{\gamma,\mathbf{n}_x} \ , \tag{57}$$

with $\mathbf{n}_x$ and $\mathbf{n}_y$ denoting unit vectors along the x and y axis, and $\mathbf{R}_{\alpha,\mathbf{n}_x}$ denotes a rotation of α degrees around axis $\mathbf{n}_x$. We can model points on the sphere in a similar way via Euler angles via

$$\mathbf{n}_{\alpha,\beta} := \mathbf{R}_{\alpha,\beta,0}\mathbf{n}_x \ . \tag{58}$$

So, with two rotation angles, any point on the sphere can be reached. In the above we set $\gamma = 0$ in the parametrisation of the rotation matrix (an element from SO(3)) that rotates the reference vector $\mathbf{n}_x$, but it should be clear that with any γ the same point $\mathbf{n}_{\alpha,\gamma}$ is reached. This means that there are many group elements in $SO(3)$ that all map $\mathbf{n}_x$ to the same point on the sphere. These elements form the subgroup $H = \mathrm{Stab}_{SO(3)}(\mathbf{n}_x)$ by which we can identify the sphere with the quotient space $S^2 \equiv SO(3)/SO(2)$.

4.2 Steerable vectors, Wigner-D matrices and irreducible representations

In Part I we defined CNNs for signal data by constraining linear layers $\mathcal{K} : \mathbb{L}_2(X) \to \mathbb{L}_2(Y)$ to be equivariant to the regular representation $\mathcal{L}_g$ of G. Now, in this Part II, we follow a similar construction. However, now we consider linear maps $\mathcal{K} : \mathbb{R}^{2l+1} \to \mathbb{R}^{2l+1}$ between a special class $(2l+1)$-dimensional vector spaces -which we call steerable vector spaces- and constrain these maps to be equivariant to the irreducible representations of G. Before we explain this in technical details, we remark the following.

Definition 4.1 (Equivalence of matrix representations)**.** Any two matrix representations $\mathbf{D}(g)$ and $\mathbf{D}'(g)$ of a group G are *equivalent* if they relate via a similarity transform

$$\mathbf{D}'(g) = \mathbf{Q}^{-1}\mathbf{D}(g)\mathbf{Q} \ ,$$

in which $\mathbf{Q}$ carries out the change of basis.

Definition 4.2 (Reducible/irreducible matrix representation)**.** A matrix representation is called reducible if it can be written as

$$\mathbf{D}(g) = \mathbf{Q}^{-1}(\mathbf{D}_1(g) \oplus \mathbf{D}_2(g))\mathbf{Q} = \mathbf{Q}^{-1} \begin{pmatrix} \mathbf{D}_1(g) & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2(g) \end{pmatrix} \mathbf{Q} \ ,$$

in which $\mathbf{Q}$ carries out a change of basis. If the matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are not reducible they are called irreducible representations (irreps).

Wigner-D matrices are irreducible representations of $SO(3)$ For SO(3) there exists a class of matrix representations, indexed with their order $l \geq 0$, which act on vector spaces of dimension $2l+1$. These representations are called Wigner-D matrices and we denote them with $\mathbf{D}^{(l)}(g)$. The use of Wigner-D matrices is motivated by the fact that any matrix representation $\mathbf{D}(g)$ of SO(3) that acts on some vector space V can be "reduced" to an equivalent block diagonal matrix representation with Wigner-D matrices along the diagonal:

$$\mathbf{D}(g) = \mathbf{Q}^{-1}(\mathbf{D}^{(l_1)}(g) \oplus \mathbf{D}^{(l_2)}(g) \oplus \dots)\mathbf{Q} = \mathbf{Q}^{-1} \begin{pmatrix} \mathbf{D}^{(l_1)}(g) & & \\ & \mathbf{D}^{(l_2)}(g) & \\ & & \ddots \end{pmatrix} \mathbf{Q} \ , \tag{59}$$

with $\mathbf{Q}$ the change of basis that makes them equivalent. The individual Wigner-D matrices themselves cannot be reduced and are hence *irreducible representations* of SO(3). Thus, since the block diagonal representations are equivalent to $\mathbf{D}$ we may as well work with them instead. This is convenient since each block, i.e., each Wigner-D matrix $\mathbf{D}^{(l_i)}$ in the diagonalization, only acts on a sub-space V_{l_1} of V. As such we can factorize $V = V_{l_1} \oplus V_{l_2} \oplus \dots$.

Definition 4.3 (Wigner-D matrix)**.** The Wigner-D matrices of type-l are the irreducible $(2l+1)$ matrix representations of $SO(3)$. A Wigner-D matrix of type l as a function of $g \in G$ will be denoted with $\mathbf{D}^{(l)}(g)$.

Definition 4.4 (Wigner-D functions)**.** The $(2l+1) \times (2l+1)$ components of the type-l Wigner-D matrices will be referred to as the type-l Wigner-D functions. The Wigner-D functions are denoted with $D^{(l)}_{mn}$ with m and n row and column index respectively.

Steerable vectors Since the sub-vector spaces V_{l_1} transform independently from one another, we will threat them as canonical objects (or channels) that deserve a special name. We will refer to these sub-spaces as steerable vector spaces.

Definition 4.5 (Steerable vector spaces and steerable vectors)**.** The $(2l+1)$-dimensional vector space on which a Wigner-D matrix of order l acts will be called *a type l steerable vector space* and is denoted with V_l. A $(2l+1)$-dimensional vector $\mathbf{v} \in V_l$ will be called a type-l vector.

E.g., a type-3 vector $\mathbf{h} \in V_3$ is transformed by $g \in SO(3)$ via $\mathbf{h} \mapsto \mathbf{D}_3(g)\mathbf{h}$. We remark that this definition is equivalent to the definition of *steerable functions* commonly used in computer vision [Freeman et al., 1991, Hel-Or and Teo, 1996] via the viewpoint that steerable vectors can be regarded as the basis coefficients of a function expanded in a spherical harmonic basis. We elicit this viewpoint in Sec. 4.3 and **??**.

Example 4.1 (Steerable vector spaces of type 0 and 1)**.** At this point we are already familiar with type-0 and type-1 steerable vector spaces. Namely, type-0 vectors $h \in V_0 = \mathbb{R}$ are scalars, which are invariant to transformations $g \in \mathrm{O}(3)$, i.e., $\mathbf{D}_0(g)h = h$. Type-1 features are vectors $\mathbf{h} \in \mathbb{R}^3$ which transform directly via the matrix representation of the group, i.e., $\mathbf{D}_1(g)\mathbf{h} = \mathbf{R}\mathbf{h}$.

Wigner-D matrices adapted to $O(3)$ The Wigner-D matrices are the irreducible representations of $SO(3)$, but we can easily adapt these representations to be suitable for $O(3)$ by including the group of reflections as a direct product. We will still refer to these representations as Wigner-D matrices in the entirety of this work, opting to avoid the distinction in favor of clarity of exposition. We further remark that explicit forms of the Wigner-D matrices can e.g. be found books such as [Sakurai and Napolitano, 2017] and their numerical implementations in code libraries such as the `e3nn` library [Geiger et al., 2021].

4.3 Fourier transform on S^2 and $SO(3)$

In Section 5 we will continue the discussion on group equivariant deep learning architectures, where we will place the content of this chapter in context. When it comes to steerable group convolutions there are two mathematical tools that

need to be understood: the *Fourier transform* on $SO(3)$ and the *Clebsch-Gordan tensor product*. Both allow us to construct operators that map from one steerable vector space to another, and we will use them in Section 5 to build steerable $SO(3)$ and $SE(3)$ group convolutions respectively. So, let us start by introducing Fourier theory on $SO(3)$, and continue with the Clebsch-Gordan product in Section 4.4.

4.3.1 Spherical Harmonics

Related to the Wigner-D matrices and their steerable vector spaces are the spherical harmonics[7]. Spherical harmonics are a class of functions on the sphere S^2, akin to the familiar circular harmonics that are best known as the 1D Fourier basis. As with a Fourier basis, spherical harmonics form an orthonormal basis for functions on S^2. In this work we use the real-valued spherical harmonics and denote them with $Y_m^{(l)} : S^2 \to \mathbb{R}$.

One can also think of spherical harmonics as functions $\tilde{Y}_m^{(l)} : SO(3) \to \mathbb{R}$ on $SO(3)$ that are invariant to a subgroup of rotations via

$$Y_m^{(l)}(\mathbf{n}_{\alpha,\beta}) = Y_m^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}\mathbf{n}_x) =: \tilde{Y}_m^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}) \ ,$$

in which we used the parametrisation for S^2 and $\mathrm{O}(3)$ given in (5.1) and (57) respectively. Then, by definition, $\tilde{Y}_m^{(l)}$ is invariant with respect to rotation angle γ, i.e.., $\forall_{\gamma\in[0,2\pi)} : \quad \tilde{Y}_m^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}) = \tilde{Y}_m^{(l)}(\mathbf{R}_{\alpha,\beta,0})$. This viewpoint of regarding the spherical harmonics as γ-invariant functions on $SO(3)$ helps us to draw the connection to the Wigner-D functions $D_{mn}^{(l)}$ that make up the $2l+1 \times 2l+1$ elements of the Wigner-D matrices. Namely, the functions in the $n=0$ column of Wigner-D matrices are also γ-invariant and, in fact, correspond (up to a normalisation factor) to the spherical harmonics via

$$Y_m^{(l)}(\mathbf{n}_{\alpha,\beta}) = \frac{1}{\sqrt{2l+1}} D_{m0}^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}) \ . \tag{60}$$

Thus, spherical harmonics $Y_m^{(l)}$ are Wigner-D functions. In fact, each column of Wigner-D functions in $\mathbf{D}^{(l)}$ is acted upon independently from one another via the Wigner-D matrix[8]. It follows that then that these column vectors are steerable vectors, as we explain next.

The embedding of a vector $\mathbf{n} \in S^2$ in spherical harmonics is equivariant
Let

$$\mathbf{a}^{(l)}(\mathbf{n}) := (Y_{-l}^{(l)}(\mathbf{n}), \dots, Y_l^{(l)}(\mathbf{n}))^T$$

be the embedding of a direction vector $\mathbf{n} \in S^2$ in spherical harmonics. Then this vector embedding is equivariant as it satisfies

$$\forall_{\tilde{\mathbf{R}}\in \mathrm{SO}(3)} \ \ \forall_{\mathbf{n}\in S^2} : \quad \mathbf{a}^{(l)}(\tilde{\mathbf{R}}\mathbf{n}) = \mathbf{D}^{(l)}(\tilde{\mathbf{R}})\mathbf{a}^{(l)}(\mathbf{n}) \ . \tag{61}$$

[7] Solutions to Laplace's equation are called harmonics. Solutions of Laplace's equation on the sphere are therefore called spherical harmonics.

[8] See $\mathrm{vec}(\mathbf{D}^{(l)}(g)\mathbf{D}^{(l)}(\tilde{g})I) = (\mathbf{I} \otimes \mathbf{D}^{(l)}(g^{-1}))\,\mathrm{vec}(\mathbf{D}^{(l)}(g))$.

This is derived by Using the S^2 and $SO(3)$ parametrization of (5.1) and (57) via

$$\begin{aligned}\mathbf{a}^{(l)}(\tilde{\mathbf{R}}\mathbf{n}_{\alpha,\beta}) &= \begin{pmatrix} Y_{-l}^{(l)}(\tilde{\mathbf{R}}\mathbf{n}_{\alpha,\beta}) \\ \vdots \\ Y_{l}^{(l)}(\tilde{\mathbf{R}}\mathbf{n}_{\alpha,\beta}) \end{pmatrix} \overset{\star}{=} \begin{pmatrix} D_{-l0}^{(l)}(\tilde{\mathbf{R}}\mathbf{R}_{\alpha,\beta,\gamma}) \\ \vdots \\ D_{l0}^{(l)}(\tilde{\mathbf{R}}\mathbf{R}_{\alpha,\beta,\gamma}) \end{pmatrix} \\ &\overset{*}{=} \mathbf{D}^{(l)}(\tilde{\mathbf{R}}) \begin{pmatrix} D_{-l0}^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}) \\ \vdots \\ D_{l0}^{(l)}(\mathbf{R}_{\alpha,\beta,\gamma}) \end{pmatrix} = \mathbf{D}^{(l)}(\mathbf{R}') \begin{pmatrix} Y_{-l}^{(l)}(\mathbf{R}'\mathbf{n}_{\alpha,\beta}) \\ \vdots \\ Y_{l}^{(l)}(\mathbf{R}'\mathbf{n}_{\alpha,\beta}) \end{pmatrix} \\ &= \mathbf{D}^{(l)}(\mathbf{R}')\mathbf{a}^{(l)}(\mathbf{n}_{\alpha,\beta}) \ ,\end{aligned}$$

where at $\overset{\star}{=}$ we used that the spherical harmonics are the central column of the Wigner-D matrices, and at $\overset{*}{=}$ we use that $\mathbf{D}^{(l)}(g)\mathbf{D}^{(l)}(\tilde{g}) = \mathbf{D}^{(l)}(g\tilde{g})$.

4.3.2 Irreducible representations as a Fourier basis

Fourier transform on S^2 Just like the 1D Fourier basis forms a complete orthonormal basis for 1D functions, the spherical harmonics $Y_m^{(l)}$ form an orthonormal basis for $\mathbb{L}_2(S^2)$, the space of square integrable functions on the sphere. In other words, any function on the sphere can be represented by a steerable vector $\mathbf{a} \in V_0 \oplus V_1 \oplus \dots$ when it is expressed in a spherical harmonic basis via

$$f(\mathbf{n}) = \sum_{l \geq 0} \sum_{m=-l}^{l} a_m^{(l)} Y_m^{(l)}(\mathbf{n}) \ . \tag{62}$$

Since spherical harmonics form an orthonormal basis, the coefficient vector $\mathbf{a}$ can directly be obtained taking $\mathbb{L}_2(S^2)$-inner products of the function f with the spherical harmonics, i.e.,

$$a_m^{(l)} = (f, Y_m^{(l)})_{\mathbb{L}_2(S^2)} = \int_{S^2} f(\mathbf{n}) Y_m^{(l)}(\mathbf{n}) \mathrm{d}\mathbf{n} \ . \tag{63}$$

Equation (63) is sometimes referred to as the Fourier transform on S^2, and Eq. (62) as the inverse spherical Fourier transform. Thus, one can identify steerable vectors $\mathbf{a}$ with functions on the sphere S^2 via the inverse spherical Fourier transform (Eq. (63). For notational convenience we will define the spherical Fourier transform as follows.

Definition 4.6 (Spherical Fourier transform)**.** Let $f \in \mathbb{L}_2(S^2)$ be a spherical signal and let $\hat{f}(l) \in \mathbb{R}^{2l+1}$ denote the vector of Fourier coefficients of order l. We may refer to l as the frequency index. The forward and inverse Fourier transform are respectively given by

$$\hat{f}(l) = \int_{S^2} f(\mathbf{n}) \underline{Y}^{(l)}(\mathbf{n}) \mathrm{d}\mathbf{n} \tag{64}$$

$$f(\mathbf{n}) = \sum_{l \geq 0} \hat{f}(l)^T \underline{Y}^{(l)}(\mathbf{n}) \,, \tag{65}$$

with $\underline{Y}^{(l)} = (Y^{(l)}_{-l}, \dots, Y^{(l)}_{l})^T \in \mathbb{L}_2(S^2)^{2l+1}$ the vector of spherical harmonics.

Exercise 4.1 (Shift property)**.** Show that the spherical Fourier transform is $SO(3)$ equivariant via

$$\widehat{\mathcal{L}_g f}(l) = \mathbf{D}^{(l)}(g) \hat{f}(l) \,.$$

Fourier transform on $SO(3)$ In order to draw a connection between regular G-CNNs and steerable G-CNNs, as we will in Section 5, it is important to understand that steerable vectors may also represent functions on the group $SO(3)$ via an $SO(3)$-Fourier transform. Namely, the full set of Wigner-D functions $D^{(l)}_{mn}$ form an orthonormal basis for $\mathbb{L}_2(SO(3))$ that allows for a Fourier transform that maps functions in $\mathbb{L}_2(SO(3))$ to steerable vectors in $V_l \oplus V_l \oplus \cdots \oplus V_l$ ($2l+1$ times) (each of the $2l_1$ columns is an independent steerable vector space). The forward and inverse Fourier transform on O(3) are respectively given by

$$\hat{f}^{(l)}_{mn} = \int_{SO(3)} f(g) D^{(l)}_{mn}(g) \mathrm{d}g \,, \tag{66}$$

$$f(g) = \sum_{l \geq 0} \sum_{m=-l}^{l} \hat{f}^{(l)}_{mn} D^{(l)}_{mn}(g^{-1}) \,, \tag{67}$$

with $\mathrm{d}g$ the Haar measure on $SO(3)$ group. We will apply the same notation as in Definition 4.6, however, now the Fourier coefficients are not vector valued but matrix valued.

Definition 4.7 ($SO(3)$ Fourier transform)**.** Let $f \in \mathbb{L}_2(SO(3))$ be a spherical signal and let $\hat{f}(l) \in \mathbb{R}^{2l+1 \times 2l+1}$ denote the matrix of Fourier coefficients of order l. The number l may be referred to as frequency index. The forward and inverse Fourier transform are respectively given by

$$\hat{f}(l) = \int_{SO(3)} f(g) \mathbf{D}^{(l)}(g) \mathrm{d}g \,, \tag{68}$$

$$f(g) = \sum_{l \geq 0} \mathrm{Tr}(\hat{f}(l) \mathbf{D}^{(l)}(g^{-1})) \,. \tag{69}$$

Generalized Fourier theorem Now that we have properly defined the Fourier transforms on S^2 and $SO(3)$, we can show that they have similar properties as the Fourier transform on $\mathbb{R}^d$. For example, as already hinted before with the equivariance property of spherical hamronic embeddings (Exercise 4.1), the $SO(3)$ Fourier transform is equivariant to the action of the group as follows.

Lemma 4.1 (Shift property)**.** Let $\mathcal{L}_g$ denote the left-regular representation of $SO(3)$ on $\mathbb{L}_2(SO(3))$ and $\hat{f}$ denote the $SO(3)$ Fourier transform $SO(3)$ (Definition 4.7). The $SO(3)$ Fourier transform is equivariant via

$$\widehat{\mathcal{L}_g f}(l) = \mathbf{D}^{(l)}(g)\hat{f}(l)\,. \tag{70}$$

Exercise 4.2. Proof Lemma 4.1. Tip apply a substitution of variables in the intregral of Eq. (68).

More importantly, we have a Fourier convolution theorem on $SO(3)$, that states that a convolution/correlation between two signals can be expressed as an element-wise multiplication of matrix-valued Fourier coefficients.

Theorem 4.1 (Convolution theorem on $SO(3)$)**.** Let $k, f \in \mathbb{L}_2(SO(3))$ and let $\hat{k}(l), \hat{f}(l)$ denote their matrix valued Fourier coefficients. Then the Fourier transform of a (group) correlation of a k with f is given by

$$\widehat{k \star f}(l) = \hat{f}(l)\hat{k}(l)^T\,. \tag{71}$$

Exercise 4.3 (Proof of Convolution Theorem)**.** Proof Theorem 4.1.

4.4 Clebsch-Gordan product and steerable linear layers

In a regular NN one maps between input and output vector spaces linearly via matrix-vector multiplication and applies non-linearities afterwards (Eq. 42). In a steerable NN one maps between steerable input and output vector spaces via the *Clebsch-Gordan tensor product* and applies non-linearities afterwards. Akin to the learnable weight matrix in regular NNs, the learnable Glebsch-Gordan tensor product will be the main workhorse of the soon to-be-introduced steerable NNs.

Clebsch-Gordan tensor product The Clebsch-Gordan (CG) tensor product allows us to map between steerale input and output spaces. While there is much to be said about tensors and tensor products in general, we here intend to focus on the core intuition. In general, a tensor product involves the multiplication between all components of two input vectors. E.g., with two vectors $\mathbf{h}_1 \in \mathbb{R}^{d_1}$ and $\mathbf{h}_2 \in \mathbb{R}^{d_2}$, the tensor product is given by

$$\mathbf{h}_1 \otimes \mathbf{h}_2 = \mathbf{h}_1\mathbf{h}_2^T = \begin{pmatrix} h_1h_1 & h_1h_2 & \dots \\ h_2h_1 & h_2h_2 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}\,,$$

which we can flatten into a $d_1 d_2$-dimensional vector via an operation which we denote with $\text{vec}(\mathbf{h}_1 \otimes \mathbf{h}_2)$.

In our steerable setting we would like to work exclusively with steerable vectors and as such we would like for any two steerable vectors $\tilde{\mathbf{h}}_1 \in V_{l_1}$ and $\tilde{\mathbf{h}}_2 \in V_{l_2}$, that the tensor product's output is again steerable with a O(3) representation $\mathbf{D}(g)$ such that the following equivariance constraint is satisfied:

$$\mathbf{D}(g)(\tilde{\mathbf{h}}_1 \otimes \tilde{\mathbf{h}}_2) = (\mathbf{D}^{(l_1)}(g)\tilde{\mathbf{h}}_1) \otimes (\mathbf{D}^{(l_2)}(g)\tilde{\mathbf{h}}_2) \ . \tag{72}$$

Via the identity $\text{vec}(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})\,\text{vec}(\mathbf{X})$, we can show that the output is indeed steerable:

$$\begin{aligned}\text{vec}\left((\mathbf{D}^{(l_1)}(g)\tilde{\mathbf{h}}_1)(\mathbf{D}^{(l_2)}(g)\tilde{\mathbf{h}}_2)^T \right) &= \text{vec}\left(\mathbf{D}^{(l_1)}(g)\tilde{\mathbf{h}}_1\tilde{\mathbf{h}}_2^T\mathbf{D}^{(l_2)T}(g) \right) \\ &= \left(\mathbf{D}^{(l_2)}(g) \otimes \mathbf{D}^{(l_1)}(g)\right) \text{vec}\left(\tilde{\mathbf{h}}_1\tilde{\mathbf{h}}_2^T \right) \ .\end{aligned}$$

The resulting vector $\text{vec}(\tilde{\mathbf{h}}_1 \otimes \tilde{\mathbf{h}}_2)$ of a tensor product with steerable vectors is again steerable by a representation $\mathbf{D}(g) = \mathbf{D}^{(l_2)}(g) \otimes \mathbf{D}^{(l_1)}(g)$. Since any matrix representation of $SO(3)$ can be reduced to a direct sum of Wigner-D matrices (see Eq. (59)), the resulting vector can be organised via a change of basis into parts that individually transform via Wigner-D matrices of different type. I.e., the tensor product can be defined such that $\tilde{\mathbf{h}} = \text{vec}(\tilde{\mathbf{h}}_1 \otimes \tilde{\mathbf{h}}_2) \in V = V_0 \oplus V_1 \oplus \ldots$, with V_l the steerable sub-vector spaces of type l.

With the CG tensor product we *directly* obtain the vector components for the steerable sub-vectors of type l.

Definition 4.8 (Clebsch-Gordan tensor product)**.** Let $\tilde{\mathbf{h}}^{(l)} \in V_l = \mathbb{R}^{2l+1}$ denote a steerable vector of type l and $h_m^{(l)}$ its components with $m = -l, -l+1, \ldots, l$. Then the Clebsch-Gordan tensor product is defined is a tensor product such tat the m-th component of the type l sub-vector of the output of the tensor product between two steerable vectors of type l_1 and l_2 is given by

$$(\tilde{\mathbf{h}}^{(l_1)} \otimes_{cg} \tilde{\mathbf{h}}^{(l_2)})_m^{(l)} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C_{(l_1,m_1)(l_2,m_2)}^{(l,m)} h_{m_1}^{(l_1)} h_{m_2}^{(l_2)} \ , \tag{73}$$

in which $C_{(l_1,m_1)(l_2,m_2)}^{(l,m)}$ are the Clebsch-Gordan coefficients. The l-th output vector $(\tilde{\mathbf{h}}^{(l_1)} \otimes_{cg} \tilde{\mathbf{h}}^{(l_2)})^{(l)} \in \mathbb{R}^{2l+1}$ is a type-l steerable vector.

The CG tensor product is a sparse tensor product, as generally many of the $C_{(l_1,m_1)(l_2,m_2)}^{(l,m)}$ components are zero. Most notably, $C_{(l_1,m_1)(l_2,m_2)}^{(l,m)} = 0$ whenever $l < |l_1 - l_2|$ or $l > l_1 + l_2$. E.g., a type-0 and a type-1 feature vector cannot create a type-2 feature vector. Well known examples of the CG product are the scalar product ($l_1 = 0, l_2 = 1, l = 1$), which takes as input a scalar and a type-1 vector to generate a type-1 vector, the dot product ($l_1 = 1, l_2 = 1, l = 0$) and cross product ($l_1 = 1, l_2 = 1, l = 1$).

Weight-parametrized Clebsch-Gordan tensor product The CG tensor product thus allows us to map two steerable input vectors to a new output vector and can furthermore be extended to define a tensor product between steerable vectors of mixed types. The CG tensor product can be parametrised by weights where the product is scaled with some weight w for each triplet of types (l_1, l_2, l) for which the CG coefficients are non-zero[9]. We indicate such CG tensor products with $\otimes_{cg}^{\mathbf{W}}$.

Definition 4.9 (Weight-parametrized Clebsch-Gordan tensor product)**.** Let $\otimes_{cg}^{\mathbf{W}}$ define the weight-parametrized Clebsch-Gordan tensor product via

$$(\tilde{\mathbf{h}}^{(l_1)} \otimes_{cg}^{\mathbf{W}} \tilde{\mathbf{h}}^{(l_2)})_m^{(l)} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} w^{(l_1,l_2,l)} C_{(l_1,m_1)(l_2,m_2)}^{(l,m)} h_{m_1}^{(l_1)} h_{m_2}^{(l_2)} , \tag{74}$$

in which $\mathbf{W}$ denotes the collection of weights $w^{(l_1,l_2,l)}$ for each path.

Each path (l_1, l_2, l) only gets a single weight $w^{(l_1,l_2,l)}$ as to ensure that the resulting steerable vector can again be interpreted as the Fourier transform of a spherical signal. In Section 5 we will make this Fourier interpretation more explicit by considering both spherical convolution as well as $SE(3)$ group convolutions.

Steerable linear layer While in principle the CG tensor product takes two steerable vectors as input, we can decide to use it with one of its input vectors "fixed". The CG tensor product can then be regarded as a linear layer that maps between two steerable vector spaces and we denote this

$$\mathbf{W}_{\tilde{\mathbf{a}}} \tilde{\mathbf{h}} := \tilde{\mathbf{h}} \otimes_{cg}^{\mathbf{W}} \tilde{\mathbf{a}} ,$$

with $\tilde{\mathbf{a}}$ the steerable vector that is considered to be fixed. With this viewpoint we can design NNs in the same way as we are used to with regular linear layers (Eq. (42)), and establish clear analogies with (point) convolution layers (Sec. ?? of the main paper).

Band-limiting We end with some notes on band-limiting (controlling the maximum frequency l). While in general the CG tensor product between two steerable vectors of type l_1 and l_2 can contain steerable vectors up to degree $l = l_1 + l_2$, one typically "band-limits" the output vector by only considering the steerable vectors up to some degree l_{max}.

The amount of interaction between the steerable sub-vectors in the hidden representations of degree $\tilde{\mathbf{h}} \in V_{L=l_f} = V_0 \oplus V_1 \oplus \dots V_{l_f}$ is furthermore determined by the maximum order l_a of the steerable vector $\tilde{\mathbf{a}}$ on which the steerable linear layer is conditioned. Namely, the CG tensor product only produces type l

[9] In terms of the `e3nn` library [Geiger et al., 2021] one then says a path exists between these 3 types.

steerable sub-vectors for $|l_f - l_a| \geq l \leq |l_f + l_a|$. For example, it is not sensible to embed positions as steerable vectors up to order $l_a = 5$ when the hidden representations are of degree $l_f = 2$; the lowest steerable vector type that can be produced with the CG product of a $l = 5$ sub-vector of $\tilde{\mathbf{a}}$ with any sub-vector of the hidden representation vector will be $l = 3$ and since the hidden representations are band-limited to a maximum type of $l_f = 2$ higher order vectors will be ignored.

5 Steerable group convolutions

We have now introduced all mathematical preliminaries for building steerable group convolutional neural networks. We will start with Section 5.1 with the S^2 and $SO(3)$ case and build steerable group convolution layers using the Fourier transform on $SO(3)$. In Section 5.2 we will then build steerable G-CNNs for $SE(3)$. Before we start, we reiterate the message that steerable group convolutions are important as they do not require a discretization of the domain $SO(3)$.

5.1 $SO(3)$ Equivariant Steerable Group Convolutions for Spherical Data

Following Theorem 3.2, i.e., the most important insight of these notes that every equivariant linear layer between signals is a group convolution, we begin by identifying the relevant cases for $SO(3)$.

The setting is as follows. We assume that our data is provided as a (sampled) function on the sphere. So our input feature maps are in $X = \mathbb{L}_2(S^2)$. We then can consider the following cases for the domains on which the represent the output feature maps:

- $Y = S^2$: We can build G-CNNs that work exclusively with spherical feature maps. As we will see, this implies that only isotropic convolution kernels can be used.
- $Y = SO(3)$: We can lift the data to $SO(3)$-feature maps. This way, there are no constraints on the convolution kernels.
- $Y = SO(3)/SO(3)$: We can pool over the entire domain of the input as to generate a single scalar number.

Before we proceed, recall from Section 4.1.2 that S^2 is a homogeneous space of $SO(3)$. Furthermore, recall the (α, β, γ) parametrization for $SO(3)$ and S^2 given respectively by

$$\mathbf{R}_{\alpha,\beta,\gamma} = \mathbf{R}_{\alpha,\mathbf{n}_x}\mathbf{R}_{\beta,\mathbf{n}_y}\mathbf{R}_{\gamma,\mathbf{n}_x} \ ,$$

and

$$\mathbf{n}_{\alpha,\beta} := \mathbf{R}_{\alpha,\beta,0}\mathbf{n}_x \ .$$

5.1.1 Steerable isotropic S^2 convolutions ($X = Y = S^2$)

These are layers of the form Eq. 48, with $\mathcal{K}$ a convolution on the sphere, given by

$$\forall_{\gamma\in SO(2)}: \quad (k \star f)(\mathbf{n}_{(\alpha,\beta)} = (\mathcal{L}_{(\alpha,\beta,\gamma)}k, f)_{\mathbb{L}_2(S^2)} \, . \tag{75}$$

The definition of Eq. (75) requires invariance to γ rotations as we want the output signal again to be a function on the sphere. We can/should guarantee

this by making the kernel symmetric around the reference axis $\mathbf{n}_x$:

$$\forall_{\alpha \in SO(2)}: \quad k(\mathbf{R}_{(\alpha,0,0)}\mathbf{n}_x) = k(\mathbf{n}_x).$$

This is a consequence of item 1 of Theorem 3.2. Namely $S^2 \equiv SO(3)/H$ with $H = \mathrm{Stab}_{SO(3)}(\mathbf{n}_x)$. The question then is, how do we parametrize k such that the constraint is satisfied? The answer is naturally given in terms of $SO(3)$ convolutions as follows.

5.1.2 Fourier-based Group convolutions in vectorized form

Recall the convolution theorem (Theorem 4.1). In the following we are going to analyze what the kernels in the Fourier domain look like, and how they can be constrained to generate either signals on S^2 or $SO(3)$. In our analyses we will focus on signals and kernels that only contain frequency $l = 1$ components and end with formalizing the case for all l.

The convolution theorem states that we can compute the convolution of k with f by computing the forward Fourier transform of input f and perform convolution as a multiplication in the Fourier domain.

$$\widehat{k \star f}(l) = \hat{f}(l)\hat{k}^T(l)\,, \tag{76}$$

with $\hat{k}$ and $\hat{f}$ respectively the Fourier representations of k. Let us further write $\hat{w}(l) \in \mathbb{R}^{2l+1 \times 2l_1}$ to denote the learnable parameters that describe the convolution kernel. In order for convenience of analyses, we will furthermore write convolution as matrix-vector multiplication, by vectorizing the Fourier coefficient matrices. We thus write Eq. (5.1) as

$$\mathrm{vec}(\widehat{k \star f})(l) = (\hat{w}(l) \otimes I)\,\mathrm{vec}(\hat{f}(l))\,, \tag{77}$$

where we use the identity $\mathrm{vec}(\mathbf{AB}) = (\mathbf{B}^T \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{A})$ for the vectorization of products of matrices. Thus, we will represent signals on $SO(3)$ via their flattened Fourier coefficients, which are steerable vectors $\mathrm{vec}(\hat{f}) \in \underbrace{V_l \oplus V_l \oplus \cdots \oplus V_l}_{2l+1 \text{ times}}$.

The simplified case of $l = 1$ To keep things intuitive we will focus on the case of signals and convolution kernels that only contain frequency $l = 1$ components. Then, the $SO(3)$ Fourier transform gives the components

$$\mathrm{vec}(\hat{f}) = \begin{pmatrix} \hat{f}_{:,-1} \\ \hat{f}_{:,0} \\ \hat{f}_{:,+1} \end{pmatrix}, \tag{78}$$

where we represent the Fourier coefficients as 9-dimensional vector in $V_1 \oplus V_1 \oplus V_2 = \mathbb{R}^9$ which contains the three columns of the $SO(3)$ Fourier coefficient matrix. Here $\hat{f}_{:,n}$ denotes the n-th column of $\hat{f}$. Recall that if the input signal

is a function on the sphere, only the central column of $\hat{f}$ is non-zero. We indicate this by making coloring it red:

$$\mathrm{vec}(\hat{f}) = \begin{pmatrix} \hat{f}_{:,-1} \\ \hat{f}_{:,0} \\ \hat{f}_{:,+1} \end{pmatrix} .$$

Now let's make Eq. (5.1) explicit:

$$\begin{pmatrix} \widehat{k \star f}_{:,-1} \\ \widehat{k \star f}_{:,0} \\ \widehat{k \star f}_{:,+1} \end{pmatrix} = \begin{pmatrix} w_{-1,-1}I & w_{-1,0}I & w_{-1,1}I \\ w_{0,-1}I & w_{0,0}I & w_{0,1}I \\ w_{1,-1}I & w_{1,0}I & w_{1,1}I \end{pmatrix} \begin{pmatrix} \hat{f}_{:,-1} \\ \hat{f}_{:,0} \\ \hat{f}_{:,+1} \end{pmatrix} , \tag{79}$$

where we have marked the parts of the weight matrix red that play no role in the multiplication in the S^2 case for which only $\hat{f}_{:,0}$ are non-zero. The color shows us that the kernel only consists of spherical harmonics and thus: if the input is in $\mathbb{L}_2(S^2)$, so will the convolution kernel be!

Eq. (80) shows us that an $SO(3)$ group convolution with a kernel $k \in \mathbb{L}_2(S^2)$ does not necessarily give us a signal on the sphere again. If we want this, then this means that the output should only have non-zero coefficients for $n = 0$. We denote this by coloring blue the weights of the kernel that induce this

$$\begin{pmatrix} \widehat{k \star f}_{:,-1} \\ \widehat{k \star f}_{:,0} \\ \widehat{k \star f}_{:,+1} \end{pmatrix} = \begin{pmatrix} w_{-1,-1}I & w_{-1,0}I & w_{-1,1}I \\ w_{0,-1}I & w_{0,0}I & w_{0,1}I \\ w_{1,-1}I & w_{1,0}I & w_{1,1}I \end{pmatrix} \begin{pmatrix} \hat{f}_{:,-1} \\ \hat{f}_{:,0} \\ \hat{f}_{:,+1} \end{pmatrix} , \tag{80}$$

Through the above analysis we conclude the following.

Lemma 5.1. Consider the cases of $SO(3)$ equivariant linear layers for signals on $X = S^2 \equiv SO(3)/SO(2)$ or $X = SO(3)$. Such convolutions can be performed via the Fourier transform on $SO(3)$

$$\widehat{k \star f}(l) = \hat{f}(l)\hat{w}^T(l)\,,$$

with $\hat{w}(l) \in \mathbb{R}^{2l+1 \times 2l_1}$ the learnable parameters of the convolution kernel. The following can be said about the parametrization of the kernels through $\hat{w}$.

- **Isotropic S^2 kernel convolutions ($X = Y = S^2$)** For isotropic kernel convolutions the kernel is parametrized by a single weight per frequency l. Namely the only possibly non-zero component is $\hat{w}_{00}(l)$ and all others have to be 0. This means that the kernels represent signals on the sphere S^2 that are symmetry around the $\mathbf{n}_x$ axis, as required by Theorem 3.2.
- **Lifting convolutions ($X = S^2$, $Y = SO(3)$)** For lifting convolutions the kernel is parametrized by $(2l+1)$ learnable weights per frequency l. The only possible non-zero weights are $\hat{w}_{:,0}$, i.e., the central column ($n = 0$) of $\hat{w}(l)$. This means that the convolution kernels are unconstrained signals on the sphere S^2.
- **Group convolution ($X = Y = SO(3)$)** Group convolutions are parametrized by $(2l+1) \times (2l+1)$ learnable weights per frequency l. The kernels represent unconstrained functions on $SO(3)$.

5.1.3 Conclusion

Convolutions on the sphere are naturally performed via the spherical, or $SO(3)$, Fourier transform. This requires no discretization at all and equivariance to rotations is exact through the application of the Wigner-D matrices. That is, the Fourier rerpresentations form steerable vector spaces that are steered by the Wigner-D matrices.

What is currently missing in these notes, is, however, a discusion on how to build deep G-CNNs with them. In particular in the context of the non-linearities σ that we can apply. Namely, in the general formulation of Eq. (48), these are applied in the signal domain which would require an inverse Fourier transform and sampling the signal on a grid. Alternatively, one could define non-linearities in the Fourier domain. I may discuss these in a future version of these notes, for now please see the literature on steerable G-CNNs for details.

5.2 $SE(3)$ Equivariant Steerable Group Convolutions

Our next case of steerable group convolutions will be for signals on $\mathbb{R}^d$ and $SE(3)$. This is where the Clebsch-Gordan tensor product comes into the picture. What will be different to the regular group convolutional setting is that now we do not treat $SE(3)$-feature maps as functions $f : SE(3) \to \mathbb{R}^N$ that map each group element to a N-dimensional feature vector in $\mathbb{R}^N$, but instead consider feature maps $f : \mathbb{R}^3 \to V$, with V a steerable vector space, i.e., we define the feature maps as vector-fields of steerable vectors. In the previous Section 5.1 we saw that steerable vectors can be treated as signals on S^2 or $SO(3)$ which allow us to treat such vector fields of steerable vectors as functions that assign to each location a function on $SO(3)$. To come to this intuition, we start off with thte case of regular $SE(3)$ convolutions with kernels expanded in a basis of spherical harmonics.

5.2.1 Steerable functions.

Before we proceed we revisit the definition of steerability as it is commonly given in the computer vision field. Through the equivalence of steerable vectors and spherical functions via the spherical Fourier transform, it is clear that our definition of *steerable vectors* coincides with the more common definition of *steerable functions*, as commonly used in computer vision [Freeman et al., 1991]. In this context, a function f is called steerable [Hel-Or and Teo, 1996] under a transformation group G if any transformation $g \in G$ on f can be written as a linear combination of a fixed, finite dimensionla vector of n basis functions $\mathbf{\Phi} = (\phi_i)_{i=1}^n$:

$$\mathcal{L}_g f = \sum_{i=1}^{n} \alpha_i(g)\phi_i = \boldsymbol{\alpha}^T(g)\mathbf{\Phi} \ , \tag{81}$$

in which $\mathcal{L}_g$ is the left regular representation of the group G that performs the transformation on f. In case of functions in a spherical harmonic basis, which we denote with

$$f_{\mathbf{a}} = \sum_{l \geq 0} \sum_{m=-l}^{l} a_m^{(l)} Y_m^l(\mathbf{n}) \, ,$$

it directly follows from Eq. (61) that they are steerable via

$$\mathcal{L}_g f_{\mathbf{a}}(\mathbf{n}) = f_{\mathbf{D}(g)\mathbf{a}}(\mathbf{n}) \ . \tag{82}$$

In terms of the steerable function definition in Eq. (81), this means that $\boldsymbol{\alpha}(g) = \mathbf{D}(g)\mathbf{a}$ and $\mathbf{\Phi} = (Y_0^{(0)}, Y_{-1}^{(1)}, Y_0^{(1)}, \dots)^T$, i.e. the set of basis functions flattened into a vector.

5.2.2 Lifting convolution ($X = \mathbb{R}^d$, $Y = SE(3)$)

The notion of steerability becomes particularly clear when viewed in the computer vision context [Freeman et al., 1991], where one may be interested in de-

tecting visual features under arbitrary rotations. Let us revisit the case of 3D cross-correlations of a kernel $k : \mathbb{R}^3 \to \mathbb{R}$ with an input feature map $f : \mathbb{R}^3 \to \mathbb{R}$:

$$f'(\mathbf{x}) = (k \star f)(\mathbf{x}) = \int_{\mathbb{R}^3} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}' \ . \tag{83}$$

In many applications, we want to detect such patterns under arbitrary rotations. For example, in 3D medical image data there is no preferred orientation and features (such as blood vessels, lesions, ...) can appear under any angle, and the same holds for particular atomic patterns in molecules. So ideally, one wants to apply the convolution kernel under all such transformations:

$$f'(\mathbf{x}, \mathbf{R}) = \int_{\mathbb{R}^3} k(\mathbf{R}^{-1}(\mathbf{x}' - \mathbf{x})) f(\mathbf{x}') \mathrm{d}\mathbf{x}' \ . \tag{84}$$

By repeating the convolutions with rotated kernels we are able to detect the presence of a certain feature at all possible angles. This describes a lifting group correlation as defined in Definition 3.2 (feature maps are lifted from $\mathbb{R}^3$ to the group SE(3)) and the subsequent layers should be defined by group convolutions (cf. Chapter 3).

Regular vs Steerable Group convolutions. In *regular group convolutional neural networks* one continues to work with such higher dimensional feature maps in which the kernels are also functions on the group. The lifting and subsequent group convolutions then all have the same form and are defined via the group action on $\mathbb{R}^3$ and group product respectively via

$$f'(g) = \int_{\mathbb{R}^3} k(g^{-1} \cdot \mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \ , \tag{85}$$

$$f'(g) = \int_{SE(3)} k(g^{-1} \cdot g') f(g') \mathrm{d}g' \ , \tag{86}$$

where $g \in \mathrm{SE}(3)$, $\mathrm{d}g$ the Haar measure on the group and where $\cdot$ in (85) and (86) respectively denote the group action on $\mathbb{R}^3$ and group product of SE(3) (cf. Section 4.1). Note that Eq. (85) is exactly the same as Eq. (84) but in different notation.

The lifting group convolution thus generates a function on the joint space of positions $\mathbb{R}^3$ and rotations SO(3). In numerical implementations, this space needs to be discretised, i.e., for a particular finite grid of rotations we want to store the results of the convolutions for each rotation $\mathbf{R}$. This approach then requires that the convolution kernel is continuous and can be sampled under all transformations. Hence, such kernels can be expanded in a continuous basis such as spherical harmonics [Weiler et al., 2018], B-splines [Bekkers, 2019] or they can be parametrised via MLPs [Finzi et al., 2020]. Alternatively, the kernels are only transformed via a sub-group of transformations in E(3) that leaves the grid on which the kernel is defined intact, as in [Worrall and Brostow, 2018, Winkels and Cohen, 2018]. An advantage of regular group convolution methods

is that normal point-wise activation functions can directly be applied to the feature maps; a down-side is that these methods are only equivariant to the sub-group on which they are discretised. When expressing the convolution kernel in terms of spherical harmonics, however, there is no need for such a discretisation at all and one can obtain the response of the convolution at any rotation $\mathbf{R}$ after convolving with the basis functions. This works as follows.

5.2.3 From regular to steerable group convolutions: kernels in a spherical harmonic basis.

Suppose a 3D convolution kernel that is expanded in a spherical harmonic basis up to degree L as follows

$$k(\mathbf{x}) = k_{\tilde{\mathbf{c}}(\|\mathbf{x}\|)}(\mathbf{x}) := \sum_{l}^{L} \sum_{m=-l}^{l} c_m^{(l)}(\|\mathbf{x}\|) Y_m^{(l)}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right) , \tag{87}$$

with $\tilde{\mathbf{c}} = (c_0^{(0)}, c_{-1}^{(1)}, c_0^{(1)} \dots)^T$ the vector of basis coefficients that can depend on $\|\mathbf{x}\|$. The coefficients can e.g. be parametrised with an MLP that takes as input $\|\mathbf{x}\|$ and returns the coefficient vector. We note that such coefficients are then $O(3)$ invariant, i.e., $\forall_{\mathbf{R}\in O(3)} : \tilde{\mathbf{c}}(\|\mathbf{R}\mathbf{x}\|) = \tilde{\mathbf{c}}(\|\mathbf{x}\|)$. Furthermore, we labelled the vector with a " ˜ " to indicate it is a steerable vector as it represents the coefficients relative to a spherical harmonic basis. It then follows (from Eq. (82)) that the kernel is steerable via

$$k(\mathbf{R}^{-1}\mathbf{x}) = k_{\mathbf{D}(\mathbf{R})\tilde{\mathbf{c}}(\|\mathbf{x}\|)}(\mathbf{x}) ,$$

i.e., via a transformation of the coefficient vector $\tilde{\mathbf{c}}$ by it $O(3)$ representation $\mathbf{D}(\mathbf{R})$.

This steerability property, together with linearity of the convolutions and basis expansion, implies that with such steerable convolution kernels we can obtain their convolutional response at any rotation directly from convolutions with the basis functions. Instead of first expanding the kernel in the basis by taking a weighted sum of basis functions with their corresponding coefficients, and only then doing the convolutions, we can change this order and first do the convolution with the basis functions and sum afterwards. In doing so we create a vector of responses $\tilde{\mathbf{f}}(\mathbf{x}) = (f_0^{(0)}(\mathbf{x}), f_{-1}^{(1)}(\mathbf{x}), f_0^{(1)}(\mathbf{x}), \dots)^T$ of which the elements are given by

$$\begin{aligned} f_m^{(l)}(\mathbf{x}) &= ((c_m^l Y_m^{(l)}) \star f^{in})(\mathbf{x}) \\ &= \int_{\mathbb{R}^3} c_m^{(l)}(\|\mathbf{x}\|) Y_m^{(l)}(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}) \mathrm{d}\mathbf{x} . \end{aligned} \tag{88}$$

Then the original $\mathbb{R}^d$ convolutional result with the kernel is obtained simply by a sum over the vector components which we denote with $\underset{l,m}{\text{SumReduce}}$ as follows

$$f'(\mathbf{x}) = \underset{l,m}{\text{SumReduce}}(\tilde{\mathbf{f}}'(\mathbf{x})) := \sum_{l}^{L} \sum_{m=-l}^{l} f_m'^{(l)}(\mathbf{x}) .$$

If one were to be interested in the rotated filter response at $\mathbf{x}$ one can first rotate the steerable vector $\tilde{\mathbf{f}}'(\mathbf{x})$ via the matrix representation $\mathbf{D}(\mathbf{R})$ and only then do the reduction. I.e., once the convolutions of (88) are done the lifting group convolution result is directly obtained via

$$f'(\mathbf{x}, \mathbf{R}) = \underset{l,m}{\text{SumReduce}}(\mathbf{D}(\mathbf{R})\mathbf{f}'(\mathbf{x})) \ . \tag{89}$$

This reduction with a multiplication with the Wigner-D matrices is in fact a point-wise inverse Fourier transform, applied to the vector $\mathbf{f}'(\mathbf{x})$ of Fourier coefficients at location $\mathbf{x}$!

5.2.4 Steerable group convolutions

When working with steerable convolution kernels one does not have to work with a grid on O(3). There are reasons to avoid working with a grid on O(3) as one cannot numerically obtain exact equivariance if the chosen grid is not a sub-group of O(3). When limiting to a discrete subgroups one can guarantee exact equivariance to the sub-group, but ideally one obtains equivariance to the entire group O(3). Steerable methods provide a way to build neural networks entirely independent of any sampling on O(3), since, as we have seen, steerable convolutions directly result in functions on the entire group O(3) via steerable vectors. That is, at each location $\mathbf{x}$ we have a steerable vector $\tilde{\mathbf{f}}(\mathbf{x})$ which represents a function on O(3) via the inverse Fourier transform given in Definition. 4.7. In fact, the sum reduction in Eq. (89) corresponds to a discrete inverse Fourier transform.

In the above example the input feature map was one that only provided a single scalar value per location $\mathbf{x}$, i.e. a type-0 steerable vector, and the output was a steerable vector field containing steerable vectors up to type L. The transition from type-0 vector field to a type-L vector field happened via tensor products with steerable vectors of spherical harmonics, and this tensor product was parametrised by the coefficients $\tilde{c}$. Suppose a convolution of a single type-l_1 steerable input field with a convolution kernel that is expanded in a spherical harmonic basis of only type l_2 via $k(\mathbf{x}) = \sum_{m=-l_2}^{l_2} w(\|\mathbf{x}\|) Y_m^{(l)}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$. The kernel can then be represented as a steerable vector field in itself as $\tilde{\mathbf{k}}(\mathbf{x}) = w(\|\mathbf{x}\|)\tilde{\mathbf{a}}(\mathbf{x})$, with $\tilde{\mathbf{a}}$ the type-l_2 spherical harmonic embedding. Such a steerable convolution maps from input feature maps $\tilde{\mathbf{f}} : \mathbb{R}^3 \to V_{l_1}$ using a convolution kernel $\tilde{\mathbf{k}} : \mathbb{R}^3 \to V_{l_2}$ to an output $\tilde{\mathbf{f}}' : \mathbb{R}^3 \to V_l$ via

$$\tilde{\mathbf{f}}'(\mathbf{x}) = \int_{\mathbb{R}^3} \tilde{\mathbf{f}}(\mathbf{x}) \otimes_{cg}^{w(\|\mathbf{x}\|)} \underline{Y}^{(l)}\left(\frac{\mathbf{x}'-\mathbf{x}}{\|\mathbf{x}'-\mathbf{x}\|}\right) \mathrm{d}\mathbf{x}' \ .$$

Here we assumed steerable feature vector fields of a single type, but in general such convolutions can map between vector fields of mixed type analogous to the standard convolutional case in CNNs where mappings occur between multi-channel type-0 vector fields.

5.2.5 Conclusion.

In conclusion, both steerable and regular group convolutions produce feature maps on the entire group E(3) and they are equivalent when the regular convolution kernel is expanded in a steerable basis. In regular group convolutions, the response is stored on a particular grid which is e.g. the Cartesian product of a regular 3D grid with a particular discretisation of O(3). In regular group convolutions, we directly index the responses with $(\mathbf{x}, \mathbf{R}) \in \mathrm{E}(3) \mapsto \mathbf{f}(\mathbf{x}, \mathbf{R})$. In steerable convolutions the filter responses $\mathbf{x} \to \tilde{\mathbf{f}}(\mathbf{x})$ are stored at each spatial grid point $\mathbf{x}$ in steerable vectors $\tilde{\mathbf{f}}(\mathbf{x})$ from which functions on the full group O(3) can be obtained via an inverse Fourier transform[10]. As such one recovers the regular representation via $(\mathbf{x}, \mathbf{R}) \mapsto \mathcal{F}^{-1}[\tilde{\mathbf{f}}(\mathbf{x})](\mathbf{R})$. Regular group convolutions can however not be perfectly equivariant to all transformations in O(3) due to discretisaton artifacts, or they are only equivariant to a discrete sub-group of O(3). Steerable group convolutions on the other hand are exactly equivariant to all transformations in O(3) via steerable representations of O(3).

[10]This Fourier transform enables working with classic point-wise activation functions that need to be applied point-wise. One cannot readily apply activation functions such as ReLU directly to the steerable coefficients, but one could sample the O(3) signal on a grid via the inverse Fourier transform, apply the activation function, and transform back into the steerable basis. This e.g. the approach taken in [**?**]. In this paper we work entirely in the steerable domain (O(3) Fourier space) and work with gated non-linearities [Weiler et al., 2018].

6 Steerable graph neural networks and Point Convolutions

Coming soon

Part III
Lie group equivariant neural networks

Coming soon

References

[Anderson et al., 2019] Anderson, B., Hy, T. S., and Kondor, R. (2019). Cormorant: Covariant molecular neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Bekkers et al., 2015] Bekkers, E., Duits, R., and Loog, M. (2015). Training of templates for object recognition in invertible orientation scores: Application to optic nerve head detection in retinal images. In Tai, X.-C., Bae, E., Chan, T. F., and Lysaker, M., editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 464–477, Cham. Springer International Publishing.

[Bekkers, 2019] Bekkers, E. J. (2019). B-spline cnns on lie groups. In *International Conference on Learning Representations*.

[Dehmamy et al., 2021] Dehmamy, N., Liu, Y., Walters, R., and Yu, R. (2021). Lie algebra convolutional neural networks with automatic symmetry extraction.

[Finzi et al., 2020] Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. (2020). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR.

[Finzi et al., 2021] Finzi, M., Welling, M., and Wilson, A. G. (2021). A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*.

[Freeman et al., 1991] Freeman, W. T., Adelson, E. H., et al. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906.

[Fuchs et al., 2020] Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. (2020). Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*.

[Geiger et al., 2021] Geiger, M., Smidt, T., M., A., Miller, B. K., Boomsma, W., Dice, B., Lapchevskyi, K., Weiler, M., Tyszkiewicz, M., Batzner, S., Frellsen,

J., Jung, N., Sanborn, S., Rackers, J., and Bailey, M. (2021). e3nn/e3nn: 2021-04-21.

[Hel-Or and Teo, 1996] Hel-Or, Y. and Teo, P. (1996). Canonical decomposition of steerable functions. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 809–816.

[Hutchinson et al., 2021] Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. (2021). Lietransformer: equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pages 4533–4543. PMLR.

[Sakurai and Napolitano, 2017] Sakurai, J. J. and Napolitano, J. (2017). *Modern Quantum Mechanics*. Cambridge University Press, 2 edition.

[Thomas et al., 2018] Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. (2018). Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*.

[Weiler et al., 2018] Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018). 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Winkels and Cohen, 2018] Winkels, M. and Cohen, T. S. (2018). 3d g-cnns for pulmonary nodule detection. In *International Conference on Medi- cal Imaging with Deep Learning (MIDL)*.

[Worrall and Brostow, 2018] Worrall, D. and Brostow, G. (2018). Cubenet: Equivariance to 3d rotation and translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 567–584.