Gauge Equivariant Convolutional Networks and the Icosahedral CNN

Taco S. Cohen * 1 Maurice Weiler * 2 Berkay Kicanaoglu * 2 Max Welling 1

Abstract

The principle of equivariance to symmetry transformations enables a theoretically grounded approach to neural network architecture design. Equivariant networks have shown excellent performance and data efficiency on vision and medical imaging problems that exhibit symmetries. Here we show how this principle can be extended beyond global symmetries to local gauge transformations. This enables the development of a very general class of convolutional neural networks on manifolds that depend only on the intrinsic geometry, and which includes many popular methods from equivariant and geometric deep learning.

We implement gauge equivariant CNNs for signals defined on the surface of the icosahedron, which provides a reasonable approximation of the sphere. By choosing to work with this very regular manifold, we are able to implement the gauge equivariant convolution using a single conv2d call, making it a highly scalable and practical alternative to Spherical CNNs. Using this method, we demonstrate substantial improvements over previous methods on the task of segmenting omnidirectional images and global climate patterns.

1. Introduction

By and large, progress in deep learning has been achieved through intuition-guided experimentation. This approach is indispensable and has led to many successes, but has not produced a deep understanding of why and when certain architectures work well. As a result, every new application requires an extensive architecture search, which comes at a significant labor and energy cost.

Figure 1. A gauge is a smoothly varying choice of tangent frame on a subset $U$ of a manifold $M$. A gauge is needed to represent geometrical quantities such as convolutional filters and feature maps (i.e. fields), but the choice of gauge is ultimately arbitrary. Hence, the network should be equivariant to gauge transformations, such as the change between red and blue gauge pictured here. Although a theory that tells us which architecture to use for any given problem is clearly out of reach, we can nevertheless come up with general principles to guide architecture search. One such rational design principle that has met with substantial empirical success (Winkels & Cohen, 2018; Zaheer et al., 2017; Lunter & Brown, 2018) maintains that network architectures should be equivariant to symmetries.

Besides the ubiquitous translation equivariant CNN, equivariant networks have been developed for sets, graphs, and homogeneous spaces like the sphere (see Sec. 3). In each case, the network is made equivariant to the global symmetries of the underlying space. However, manifolds do not in general have global symmetries, and so it is not obvious how one might develop equivariant CNNs for them.

General manifolds do however have local gauge symmetries, and as we will show in this paper, taking these into account is not just useful but necessary if one wishes to build manifold CNNs that depend only on the intrinsic geometry. To this end, we define a convolution-like operation on general manifolds $M$ that is equivariant to local gauge transformations (Fig. 1). This gauge equivariant convolution takes as input a number of feature fields on $M$ of various types (analogous to matter fields in physics), and produces as output new feature fields. Each field is represented by a number of feature maps, whose activations are interpreted as the coefficients of a geometrical object (e.g. scalar, vector, tensor, etc.) relative to a spatially varying frame (i.e. gauge). The network is constructed such that if the gauge is changed,
2. Gauge Equivariant Networks

Consider the problem of generalizing the classical convolution of two planar signals (e.g. a feature map and a filter) to signals defined on a manifold \( M \). The first and most natural idea comes from thinking of planar convolution in terms of shifting a filter over a feature map. Observing that shifts are symmetries of the plane (mapping the plane onto itself while preserving its structure), one is led to the idea of transforming a filter on \( M \) by the symmetries of \( M \). For instance, replacing shifts of the plane by rotations of the sphere, one obtains Spherical CNNs (Cohen et al., 2018b).

This approach works for any homogeneous space, where by definition it is possible to move from any point \( p \in M \) to any other point \( q \in M \) using an appropriate symmetry transformation (Kondor & Trivedi, 2018; Cohen et al., 2018c:a). On less symmetrical manifolds however, it may not be possible to move the filter from any point to any other point by symmetry transformations. Hence, transforming filters by symmetry transformations will in general not provide a recipe for weight sharing between filters at all points in \( M \).

Instead of symmetries, one can move the filter by parallel transport (Schonsheck et al., 2018), but as shown in Fig. 2, this leaves an ambiguity in the filter orientation, because parallel transport is path dependent. This can be addressed by using only rotation invariant filters (Boscaini et al., 2015; Bruna et al., 2014), albeit at the cost of limiting expressivity.

The key issue is that on a manifold, there is no preferred gauge (tangent frame), relative to which we can position our measurement apparatus (i.e. filters), and relative to which we can describe measurements (i.e. responses). We must choose a gauge in order to numerically represent geometrical quantities and perform computations, but since it is arbitrary, the computations should be independent of it. This does not mean however that the coefficients of the feature vectors should be invariant to gauge transformations, but rather that the feature vector itself should be invariant. That is, a gauge transformation leads to a change of basis \( e_i \mapsto \tilde{e}_i \) of the feature space (fiber) at \( p \in M \), so the feature vector coefficients \( f_i \) should change equivariantly to ensure that the vector \( \sum_i f_i e_i = \sum_i \tilde{f}_i \tilde{e}_i \) itself is unchanged.

Before showing how this is achieved, we note that on non-parallelizable manifolds such as the sphere, it is not possible to choose a smooth global gauge. For instance, if we extend the blue gauge pictured in Fig. 1 to the whole sphere, we will inevitably create a singularity where the gauge changes abruptly. Hence, in order to make the math work smoothly, it is standard practice in gauge theory to work with multiple gauges defined on overlapping charts, as in Fig. 1.

The basic idea of gauge equivariant convolution is as follows. Lacking alternative options, we choose arbitrarily a smooth local gauge on subsets \( U \subset M \) (e.g. the red or blue gauge in Fig. 1). We can then position a filter at each point \( p \in U \), defining its orientation relative to the gauge. Then, we match an input feature map against the filter at \( p \) to obtain the value of the output feature map at \( p \). For the output to transform equivariantly, certain linear constraints are placed on the convolution kernel. We will now define this formally.

2.1. Gauges, Transformations, and Exponential Maps

We define a gauge as a position-dependent invertible linear map \( w_p : \mathbb{R}^d \to T_p M \), where \( T_p M \) is the tangent space of \( M \) at \( p \). This determines a frame \( w_p(e_1), \ldots, w_p(e_d) \) in \( T_p M \), where \( \{e_i\} \) is the standard frame of \( \mathbb{R}^d \).
A gauge transformation (Fig. 1) is a position-dependent change of frame, which can be described by maps \( g_p \in \text{GL}(d, \mathbb{R}) \) (the group of invertible \( d \times d \) matrices). As indicated by the subscript, the transformation \( g_p \) depends on the position \( p \in U \subset M \). To change the frame, simply compose \( w_p \) with \( g_p \), i.e. \( w_p \rightarrow w_p g_p \). It follows that component vectors \( v \in \mathbb{R}^d \) transform as \( v \rightarrow g_p^{-1} v \), so that the vector \((w_p g_p)(g_p^{-1} v) = w_p v \in T_p M\) itself is invariant.

If we derive our gauge from a coordinate system for \( M \) (as shown in Fig. 1), then a change of coordinates leads to a gauge transformation (\( g_p \) being the Jacobian of the coordinate transformation at \( p \)). But we can also choose a gauge \( w_p \) independent of any coordinate system.

It is often useful to restrict the kinds of frames we consider, for example to only allow right-handed or orthogonal frames. Such restrictions limit the kinds of gauge transformations we can consider. For instance, if we allow only right-handed frames, \( g_p \) should have positive determinant (i.e. \( g_p \in \text{GL}^+(d, \mathbb{R}) \)), so that it does not reverse the orientation. If in addition we allow only orthogonal frames, \( g_p \) must be a rotation, i.e. \( g_p \in \text{SO}(d) \).

In mathematical terms, \( G = \text{GL}(d, \mathbb{R}) \) is called the structure group of the theory, and limiting the kinds of frames we consider corresponds to a reduction of the structure group (Husemoller, 1994). Each reduction corresponds to some extra structure that is preserved, such as an orientation (\( \text{GL}^+(d, \mathbb{R}) \)) or Riemannian metric (\( \text{SO}(d) \)). In the Icosahedral CNN (Fig. 4), we will want to preserve the hexagonal grid structure, which corresponds to a restriction to grid-aligned frames and a reduction of the structure group to \( G = C_6 \), the group of planar rotations by integer multiples of \( 2\pi/6 \). For the rest of this section, we will work in the Riemannian setting, i.e. use \( G = \text{SO}(d) \).

Before we can define gauge equivariant convolution, we will need the exponential map, which gives a convenient parameterization of the local neighborhood of \( p \in M \). This map \( \exp_p : T_p M \rightarrow M \) takes a tangent vector \( w \in T_p M \), follows the geodesic (shortest curve) in the direction of \( w \) with speed \( ||w|| \) for one unit of time, to arrive at a point \( q = \exp_p w \) (see Fig. 3, (Lee)).

### 2.2. Gauge Equivariant Convolution: Scalar Fields

Having defined gauges, gauge transformations, and the exponential map, we are now ready to define gauge equivariant convolution. We begin with scalar input and output fields.

We define a filter as a locally supported function \( K : \mathbb{R}^d \rightarrow \mathbb{R} \), where \( \mathbb{R}^d \) may be identified with \( T_p M \) via the gauge \( w_p \). Then, writing \( f_p = \exp_p w_p(v) \) for \( v \in \mathbb{R}^d \), we define the scalar convolution of \( K \) and \( f : M \rightarrow \mathbb{R} \) at \( p \) as follows:

\[
(K \ast f)(p) = \int_{\mathbb{R}^d} K(v) f_f(v) dv.
\]

The gauge was chosen arbitrarily, so we must consider what happens if we change it. Since the filter \( K : \mathbb{R}^d \rightarrow \mathbb{R} \) is a function of a coordinate vector \( v \in \mathbb{R}^d \), and \( v \) gets rotated by gauge transformations, the effect of a gauge transformation is a position-dependent rotation of the filters. For the convolution output to be called a scalar field, it has to be invariant to gauge transformations (i.e. \( v \rightarrow g_p^{-1} v \) and \( w_p \rightarrow w_p g_p \)). The only way to make \((K \ast f)(p)\) (Eq. 1) invariant to rotations of the filter, is to make the filter rotation-invariant:

\[
\forall g \in G : K(g^{-1} v) = K(v)
\]

Thus, to map a scalar input field to a scalar output field in a gauge equivariant manner, we need to use rotationally symmetric filters. Some geometric deep learning methods, as well as graph CNNs do indeed use isotropic filters. However, this is very limiting and as we will now show, unnecessary if one considers non-scalar feature fields.

### 2.3. Feature Fields

Intuitively, a field is an assignment of some geometrical quantity (feature vector) \( f(p) \) of the same type to each point \( p \in M \). The type of a quantity is determined by its transformation behaviour under gauge transformations. For instance, the word vector field is reserved for a field of tangent vectors \( v \), that transform like \( v(p) \rightarrow g_p^{-1} v(p) \) as we saw before. It is important to note that \( f(p) \) is an element of a vector space (“fiber”) \( F_p \simeq \mathbb{R}^C \) attached to \( p \in M \) (e.g. the tangent space \( T_p M \)). All \( F_p \) are similar to a canonical feature space \( \mathbb{R}^C \), but \( f \) can only be considered a function \( U \rightarrow \mathbb{R}^C \) locally, after we have chosen a gauge, because there is no canonical way to identify all feature spaces \( F_p \).

In the general case, the transformation behaviour of a \( C \)-dimensional geometrical quantity is described by a representation of the structure group \( G \). This is a mapping \( \rho : G \rightarrow \text{GL}(C, \mathbb{R}) \) that satisfies \( \rho(g h) = \rho(g) \rho(h) \), where \( g h \) denotes the composition of transformations in \( G \), and \( \rho(g) \rho(h) \) denotes multiplication of \( C \times C \) matrices \( \rho(g) \) and \( \rho(h) \). The simplest examples are the trivial representation \( \rho(g) = 1 \) which describes the transformation behaviour of scalars, and \( \rho(g) = g \), which describes the transformation behaviour of (tangent) vectors. A field \( f \) that transforms like \( f(p) \rightarrow \rho(g_p^{-1}) f(p) \) will be called a \( \rho \)-field.
In Section 4 on Icosahedral CNNs, we will consider one more type of representation, namely the regular representation of \( C_6 \). The group \( C_6 \) can be described as the 6 planar rotations by \( k \cdot 2\pi/6 \), or as integers \( k \) with addition mod 6. Features that transform like the regular representation of \( C_6 \) are 6-dimensional, with one component for each rotation. One can obtain a regular feature by taking a filter at \( p \), rotating it by \( k \cdot 2\pi/6 \) for \( k = 0, \ldots, 5 \), and matching each rotated filter against the input signal. When the gauge is changed, the filter and all rotated copies are rotated, and so the components of a regular \( C_6 \) feature are cyclically shifted. Hence, \( \rho \) is a \( 6 \times 6 \) cyclic permutation matrix that shifts the coordinates by \( k' \) steps for \( g = k' \cdot 2\pi/6 \). Further examples of representations \( \rho \) that are useful in convolutional networks may be found in (Cohen \\& Welling, 2017; Weiler et al., 2018a; Thomas et al., 2018; Hy et al., 2018).

### 2.4. Gauge Equivariant Convolution: General Fields

Now consider a stack of \( C_m \) input feature maps on \( M \), which represents a \( C_m \)-dimensional \( \rho_m \)-field (e.g. \( C_m = 1 \) for a single scalar, \( C_m = d \) for a vector, \( C_m = 6 \) for a regular \( C_6 \) feature, or any multiple of these, etc.). We will define a convolution operation that takes such a field and produces as output a \( C_{out} \)-dimensional \( \rho_{out} \)-field. For this we need a filter bank with \( C_{out} \) output channels and \( C_{in} \) input channels, which we will describe mathematically as a matrix-valued kernel \( K : \mathbb{R}^d \rightarrow \mathbb{R}^{C_{out} \times C_{in}} \).

We can think of \( K(v) \) as a linear map from the input feature space (“fiber”) at \( p \) to the output feature space at \( p \), these spaces being identified with \( \mathbb{R}^{C_{in}} \) resp. \( \mathbb{R}^{C_{out}} \) by the choice of gauge \( w_p \), at \( p \). This suggests that we need to modify Eq. 1 to make sure that the kernel matrix \( K(v) \) is multiplied by a feature vector at \( p \), not one at \( q \). This is achieved by transporting \( f(q) \) to \( p \) along the unique\(^1 \) geodesic connecting them, before multiplying by \( K(v) \).

As \( f(q) \) is transported to \( p \), it undergoes a transformation which will be denoted \( g_{p \rightarrow q} \in G \) (see Fig. 2). This transformation acts on the feature vector \( f(q) \in \mathbb{R}^{C_{in}} \) via the representation \( \rho_m(g_{p \rightarrow q}) \in \mathbb{R}^{C_{in} \times C_{in}} \). Thus, we obtain the generalized form of Eq. 1 for general fields:

\[
(K \ast f)(p) = \int_{\mathbb{R}^d} K(v) \rho_m(g_{p \rightarrow q}) f(q) dv.
\]  

Under a gauge transformation, we have:

\[
\begin{align*}
\forall v \in \mathbb{R}^d : & \quad v \mapsto g_p^{-1} v, \quad f(q) \mapsto \rho_m(g_p^{-1}) f(q), \\
\forall w_p \in \mathbb{R}^{C_{in}} : & \quad w_p \mapsto g_{p \rightarrow q} w_p, \quad g_{p \rightarrow q} \mapsto g_p^{-1} g_{p \rightarrow q} g_q.
\end{align*}
\]

For \( K \ast f \) to be well defined as a \( \rho_{out} \)-field, we want it to transform like \( (K \ast f)(p) \mapsto \rho_{out}(g_p^{-1})(K \ast f)(p) \). Or, in other words, \( * \) should be gauge equivariant. This will be the case if and only if \( K \) satisfies

\[
\forall g \in G : K(g^{-1} v) = \rho_{out}(g^{-1}) K(v) \rho_m(g).
\]  

This concludes our presentation of the general case. A gauge equivariant \( \rho_1 \rightarrow \rho_2 \) convolution on \( M \) is defined relative to a local gauge by Eq. 3, where the kernel satisfies the equivariance constraint of Eq. 5. By defining gauges on local charts \( U_i \subset M \) that cover \( M \) and convolving inside each one, we automatically get a globally well-defined operation, because switching charts corresponds to a gauge transformation (Fig. 1), and the convolution is gauge equivariant.

### 2.5. Locally Flat Spaces

On flat regions of the manifold, the exponential parametrization can be simplified to \( \varphi(\exp_p w_p(v)) = \varphi(p) + v \) if we use an appropriate local coordinate \( \varphi(p) \in \mathbb{R}^d \) of \( p \in M \). Moreover, in such a flat chart, parallel transport is trivial, i.e. \( g_{p \rightarrow q} \) equals the identity. Thus, on a flat region, our convolution boils down to a standard convolution / correlation:

\[
(K \ast f)(x) = \int_{\mathbb{R}^d} K(v) f(x + v) dv.
\]

Moreover, we can recover group convolutions, spherical convolutions, and convolution on other homogeneous spaces as special cases as well (see supplementary material).

### 3. Related work

**Equivariant Deep Learning**

Equivariant networks have been proposed for permutation-equivariant analysis and prediction of sets (Zaheer et al., 2017; Hartford et al., 2018), graphs (Kondor et al., 2018b; Hy et al., 2018; Maron et al., 2019), translations and rotations of the plane and 3D space (Oyallon \\& Mallat, 2015; Cohen \\& Welling, 2016; 2017; Marcos et al., 2017; Weiler et al., 2018b;a; Worrall et al., 2017; Worrall \\& Brostow, 2018; Winkels \\& Cohen, 2018; Veeeling et al., 2018; Thomas et al., 2018; Bekkers et al., 2018; Hoogeboom et al., 2018), and the sphere (see below). Ravanbakhsh et al. (2017) studied finite group equivariance. Equivariant CNNs on homogeneous spaces were studied by (Kondor \\& Trivedi, 2018) (scalar fields) and (Cohen et al., 2018c;a) (general fields). In this paper we generalize G-CNNs from homogeneous spaces to general manifolds.

---

\(^1\)For points that are close enough, there is always a unique geodesic. Since the kernel has local support, \( p \) and \( q \) will be close for all non-zero terms.
Gauge Equivariant CNNs

4. Icosahedral CNNs

In this section we will describe a concrete method for performing gauge equivariant convolution on the icosahedron. The very special shape of this manifold makes it possible to implement gauge equivariant convolution in a way that is both numerically convenient (no interpolation is required), and computationally efficient (the heavy lifting is done by a single conv2d call).

4.1. The Icosahedron

The icosahedron is a regular solid with 20 faces, 30 edges, and 12 vertices (see Fig. 4, left). It has 60 rotational symmetries. This symmetry group will be denoted $I$.

4.2. The Hexagonal Grid

Whereas general manifolds, and even spheres, do not admit completely regular and symmetrical pixellations, we can define an almost perfectly regular grid of pixels on the icosahedron. This grid is constructed through a sequence of grid-refinement steps. We begin with a grid $H_0$ consisting of the corners of the icosahedron itself. Then, for each triangular face, we subdivide it into 4 smaller triangles, thus introducing 3 new points on the center of the edges of the original triangle. This process is repeated $r$ times to obtain a grid $H_r$ with $N = 5 \times 2^{2r+1} + 2$ points (Fig. 4, left).

Each grid point (pixel) in the grid has 6 neighbours, except for the corners of the icosahedron, which have 5. Thus, one can think of the non-corner grid points as hexagonal pixels, and the corner points as pentagonal pixels.

Notice that the grid $H_r$ is perfectly symmetrical, which means that if we apply an icosahedral symmetry $g \in I$ to a point $p \in H_r$, we will always land on another grid point, i.e. $gp \in H_r$. Thus, in addition to talking about gauge equivariance, for this manifold $I$ grid, we can also talk about (exact) equivariance to global transformations (3D rotations in $I$). Because these global symmetries act by permuting the pixels and changing the gauge, one can see that a gauge equivariant network is automatically equivariant to global transformations. This will be demonstrated in Section 5.

4.3. The Atlas of Charts

We define an atlas consisting of 5 overlapping charts on the icosahedron, as shown in Fig. 4. Each chart is an invertible map $\varphi_i : U_i \rightarrow V_i$, where $U_i \subset H_r \subset M$ and $V_i \subset \mathbb{Z}^2$. The regions $U_i$ and $V_i$ are shown in Fig. 4. The maps themselves are linear on faces, and defined by hard-coded correspondences $\varphi_i(c_j) = x_j$ between the corner points $c_j$ in $H_r$ and points $x_j$ in the planar grid $\mathbb{Z}^2$.

2As an abstract group, $I \simeq A5$ (the alternating group A5), but we use $I$ to emphasize that it is realized by a set of 3D rotations.
We divide the charts into an exterior output at each interior pixel (assuming a hexagonal Jacobian of $\varphi$ the corresponding frames on the icosahedron (obtained by numerically constant and equal in both charts $p \in U_i \cap U_j$ (see Fig. 1).

For the particular atlas defined in Sec. 4.3, the gauge transformations $g_{ij}(p)$ are always elements of the group $C_6$ (i.e. rotations by $k \cdot 2\pi/6$, so $G = C_6$ and we have a $C_6$-atlas.

### 4.5. The Signal Representation

A stack of $C$ feature fields is represented as an array of shape $(B, C, R, 5, H, W)$, where $B$ is the batch size, $C$ the number of fields, $R$ is the dimension of the fields ($R = 1$ for scalars and $R = 6$ for regular features), $5$ is the number of charts, and $H, W$ are the height and width of each local chart $(H = 2r + 2$ and $W = 2^{r+1} + 2$ at resolution $r$, including a $1$-pixel padding region on each side, see Fig. 4). We can always reshape such an array to shape $(B, CR, 5H, W)$, resulting in a 4D array that can be viewed as a stack of $CR$ rectangular feature maps of shape $(5H, W)$. Such an array can be input to conv2d.

### 4.6. Gauge Equivariant Icosahedral Convolution

Gauge equivariant convolution on the icosahedron is implemented in three steps: G-Padding, kernel expansion, and 2d convolution / HexaConv (Hoogeboom et al., 2018).

#### 4.6.1. G-Padding

In a standard CNN, we can only compute a valid convolution output at positions $(x, y)$ where the filter fits inside the input image in its entirety. If the output is to be of the same size as the input, one uses zero padding. Likewise, the Ico-Conv requires padding, only now the padding border $V_i$ of a chart consists of pixels that are also represented in the interior of another chart (Sec. 4.3). So instead of zero padding, we copy the pixels from the neighbouring chart. We always use hexagonal filters with 1 ring, which can be represented as a $3 \times 3$ filter on a square grid, so we pad by 1 pixel.

As explained in Sec. 4.4, when transitioning between charts one may have to perform a gauge transformation on the features. Since scalars are invariant quantities, transition padding amounts to a simple copy in this case. Regular $C_6$ features (having 6 orientation channels) transform by cyclic shifts $\rho(g_{ij}(p))$ (Sec. 2.3), where $g_{ij} \in \{+1, 0, -1\} \cdot 2\pi/6$ (Fig. 4), so we must cyclically shift the channels up or down before copying to get the correct coefficients in the new

Figure 4. The Icosahedron with grid $H_r$ for $r = 2$ (left). We define 5 overlapping charts that cover the grid (center). Chart $V_5$ is highlighted in gray (left). Colored edges that appear in multiple charts are to be identified. In each chart, we define the gauge by the standard axis aligned basis vectors $e_1, e_2 \in V_i$. For points $p \in U_i \cap U_j$, the transition between charts involves a change of gauge, shown as $+1 \cdot 2\pi/6$ and $-1 \cdot 2\pi/6$ (elements of $G = C_6$). On the right we show how the signal is represented in a padded array of shape $5 \cdot (2^r + 2) \times (2^{r+1} + 2)$.

Each chart covers all the points in 4 triangular faces of the icosahedron. Together, the 5 charts cover all 20 faces of the icosahedron.

We divide the charts into an exterior $\overline{V_i} \subset V_i$, consisting of border pixels, and an interior $V_i^o \subset V_i$, consisting of pixels whose neighbours are all contained in chart $i$. In order to ensure that every pixel in $H_r$ (except for the 12 corners) is contained in the interior of some chart, we add a strip of pixels to the left and bottom of each chart, as shown in Fig. 4 (center). Then the interior of each chart (plus two exterior corners) has a nice rectangular shape $2^r \times 2^{r+1}$, and every non-corner is contained in exactly one interior $V_i^o$.

So if we know the values of the field in the interior of each chart, we know the whole field (except for the corners, which we ignore). However, in order to compute a valid convolution output at each interior pixel (assuming a hexagonal filter with one ring, i.e. a $3 \times 3$ masked filter), we will still need the exterior pixels to be filled in as well (introducing a small amount of redundancy). See Sec. 4.6.1.

### 4.4. The Gauge

For the purpose of computation, we fix a convenient gauge in each chart. This gauge is defined in each $V_i$ as the constant orthogonal frame $e_1 = (1, 0), e_2 = (0, 1)$, aligned with the $x$ and $y$ direction of the plane (just like the red and blue gauge in Fig. 1). When mapped to the icosahedron via (the Jacobian of) $\varphi_i^{-1}$, the resulting frames are aligned with the grid, and the basis vectors make an angle of $2 \cdot 2\pi/6$.

Some pixels $p \in U_i \cap U_j$ are covered by multiple charts. Although the local frames $e_1 = (1, 0), e_2 = (0, 1)$ are numerically constant and equal in both charts $V_i$ and $V_j$, the corresponding frames on the icosahedron (obtained by pushing them though $\varphi_i^{-1}$ and $\varphi_j^{-1}$) may not be the same. In other words, when switching from chart $i$ to chart $j$, there may be a gauge transformation $g_{ij}(p)$, which rotates the frame at $p \in U_i \cap U_j$ (see Fig. 1).

Figure 5. G-Padding (scalar signal)
4.6.2. WEIGHT SHARING & KERNEL EXPANSION

For the convolution to be gauge equivariant, the kernel must satisfy Eq. 5. The kernel \( K : \mathbb{R}^2 \rightarrow \mathbb{R}^{H \times W \times C_{in}} \) is stored in an array of shape \( (R_{out} C_{in} \times R_{in} C_{in}, 3, 3) \), with the top-right and bottom-left pixel of each \( 3 \times 3 \) filter fixed at zero so that it corresponds to a 1-ring hexagonal kernel.

Eq. 5 says that if we transform the input channels (columns) by \( \rho_{in}(g) \) and the output channels (rows) by \( \rho_{out}(g) \), the result should equal the original kernel where each channel is rotated by \( g \in C_6 \). This will be the case if we use the weight-sharing scheme shown in Fig. 6.

4.6.3. COMPLETE ALGORITHM

The complete algorithm can be summarized as

\[
\text{GConv}(f, w) = \text{conv2d}((\text{GPad}(f), \text{expand}(w))).
\]

Where \( f \) and \( \text{GPad}(f) \) both have shape \( (B, C_{in} R_{in}, 5H, W) \), the weights \( w \) have shape \( (C_{out}, C_{in} R_{in}, 7) \), and \( \text{expand}(w) \) has shape \( (C_{out} R_{out}, C_{in} R_{in}, 3, 3) \). The output of GConv has shape \( (B, C_{out} R_{out}, 5H, W) \).

On the flat faces, being described by one of the charts, this algorithm coincides exactly with the hexagonal regular convolution introduced in (Hoogeboom et al., 2018). The non-zero curvature of the icosahedron requires us to do the additional step of padding between different charts.

5. Experiments

5.1. IcoMNIST

In order to validate our implementation, highlight the potential benefits of our method, and determine the necessity of each part of the algorithm, we perform a number of experiments with the MNIST dataset, projected to the icosahedron.

We generate three different versions of the training and test sets, differing in the transformations applied to the data. In the N condition, No rotations are applied to the data. In the I condition, we apply all 60 icosahedral symmetries (rotations) to each digit. Finally, in the R condition, we apply 60 random continuous rotations \( g \in \text{SO}(3) \) to each digit before projecting. All signals are represented as explained in Sec. 4.5 / Fig. 4 (right), using resolution \( r = 4 \), i.e. as an array of shape \( (1, 5 \cdot (16 + 2), 32 + 2) \).

Our main model consists of one gauge equivariant scalar-to-regular (S2R) convolution layer, followed by 6 regular-to-regular (R2R) layers and 3 FC layers (see Supp. Mat. for architectural details). We also evaluate a model that uses only S2R convolution layers, followed by orientation pooling (a max over the 6 orientation channels of each regular feature, thus mapping a regular feature to a scalar), as in (Masci et al., 2015). Finally, we consider a model that uses only rotation-invariant filters, i.e. scalar-to-scalar (S2S) convolutions, similar to standard graph CNNs (Boscaini et al., 2015; Kipf & Welling, 2017). We also compare to the fully SO(3)-equivariant but computationally costly Spherical CNN (S2CNN). See supp. mat. for architectural details and computational complexity analysis.

In addition, we perform an ablation study where we disable each part of the algorithm. The first baseline is obtained from the full R2R network by disabling gauge padding (Sec. 4.6.1), and is called the No Pad (NP) network. In the second baseline, we disable the kernel Expansion (Sec. 4.6.2), yielding the NE condition. The third baseline, called NP+NE uses neither gauge padding nor kernel expansion, and amounts to a standard CNN applied to the same input representation. We adapt the number of channels so that all networks have roughly the same number of parameters.

<table>
<thead>
<tr>
<th>Arch.</th>
<th>N/N</th>
<th>N/I</th>
<th>N/R</th>
<th>I/I</th>
<th>I/R</th>
<th>R/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP+NE</td>
<td>99.29</td>
<td>99.50</td>
<td>16.20</td>
<td>98.52</td>
<td>47.77</td>
<td>94.19</td>
</tr>
<tr>
<td>NE</td>
<td>99.42</td>
<td>25.41</td>
<td>17.85</td>
<td>98.67</td>
<td>60.74</td>
<td>96.83</td>
</tr>
<tr>
<td>NP</td>
<td>99.27</td>
<td>36.76</td>
<td>21.4</td>
<td>98.99</td>
<td>61.62</td>
<td>97.87</td>
</tr>
<tr>
<td>S2S</td>
<td>97.81</td>
<td>97.81</td>
<td>55.64</td>
<td>97.72</td>
<td>58.37</td>
<td>89.92</td>
</tr>
<tr>
<td>S2R</td>
<td>98.99</td>
<td>98.99</td>
<td>59.76</td>
<td>98.62</td>
<td>55.57</td>
<td>98.74</td>
</tr>
<tr>
<td>R2R</td>
<td>99.43</td>
<td>99.43</td>
<td>69.99</td>
<td>99.38</td>
<td>66.26</td>
<td>99.31</td>
</tr>
</tbody>
</table>

Table 1. IcoMNIST test accuracy (%) for different architectures and train / test conditions (averaged over 3 runs). See text for explanation of labels.
As shown in Table 1, icosahedral CNNs achieve excellent performance with a test accuracy of up to 99.43%, which is a strong result even on planar MNIST, for non-augmented and non-ensembled models. The full R2R model performs best in all conditions (though not significantly in the N/N condition), showing that both gauge padding and kernel expansion are necessary, and that our general (R2R) formulation works better in practice than using scalar fields (S2S or S2R). We notice also that non-equivariant models (NP+NE, NP, NE) do not generalize well to transformed data, a problem that is only partly solved by data augmentation. On the other hand, the models S2S, S2R, and R2R are exactly equivariant to symmetries $g \in I$, and so generalize perfectly to $I$-transformed test data, even when these were not seen during training. None of the models automatically generalize to continuously rotated inputs (R), but the equivariant models are closer, and can get even closer (> 99%) when using SO(3) data augmentation during training. The fully SO(3)-equivariant S2CNN scores slightly worse than R2R, except in N/R and I/R, as expected. The slight decrease in performance of S2CNN for rotated training conditions is likely due to the fact that it has lower grid resolution near the equator. We note that the S2CNN is slower and less scalable than Ico CNNs (see supp. mat.).

### 5.2. Climate Pattern Segmentation

We evaluate our method on the climate pattern segmentation task proposed by Mudigonda et al. (2017). The goal is to segment extreme weather events (Atmospheric Rivers (AR) and Tropical Cyclones (TC)) in climate simulation data.

We use the exact same data and evaluation methodology as (Jiang et al., 2018). The preprocessed data as released by (Jiang et al., 2018) consists of 16-channel spherical images at resolution $r = 5$, which we reinterpret as icosahedral signals (introducing slight distortion). See (Mudigonda et al., 2017) for a detailed description of the data.

We compare an R2R and S2R model (details in Supp. Mat.). As shown in Table 2, our models outperform both competing methods in terms of per-class and mean accuracy. The difference between our R2R and S2R model seems small in terms of accuracy, but when evaluated in terms of mean average precision (a more appropriate evaluation metric for segmentation tasks), the R2R model clearly outperforms.

### 5.3. Stanford 2D-3D-S

For our final experiment, we evaluate icosahedral CNNs on the 2D-3D-S dataset (Armeni et al., 2017), which consists of 1413 omnidirectional RGB+D images with pixelwise semantic labels in 13 classes. Following Jiang et al. (2018), we sample the data on a grid of resolution $r = 5$ using bilinear interpolation, while using nearest-neighbour interpolation for the labels. Evaluation is performed by mean intersection over union (mIoU) and pixel accuracy (mAcc).

The network architecture is a residual U-Net (Ronneberger et al., 2015; He et al., 2016) with R2R convolutions. The network consists of a downsampling and upsampling network. The downsampling network takes as input a signal at resolution $r = 5$ and outputs feature maps at resolutions $r = 4, \ldots, 1$, with 8, 16, 32 and 64 channels. The upsampling network is the reverse of this. We pool over orientation channels right before applying softmax.

As shown in table 3, our method outperforms the method of (Jiang et al., 2018), which in turn greatly outperforms standard planar methods such as U-Net on this dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>mAcc</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jiang et al. (2018)</td>
<td>0.547</td>
<td>0.383</td>
</tr>
<tr>
<td>Ours (R2R-U-Net)</td>
<td>0.559</td>
<td>0.394</td>
</tr>
</tbody>
</table>

Table 3. Mean accuracy and intersection over union for 2D-3D-S omnidirectional segmentation task.

### 6. Conclusion

In this paper we have presented the general theory of gauge equivariant convolutional networks on manifolds, and demonstrated their utility in a special case: learning with spherical signals using the icosahedral CNN. We have demonstrated that this method performs well on a range of different problems and is highly scalable.

Although we have only touched on the connections to physics and geometry, there are indeed interesting connections, which we plan to elaborate on in the future. From the perspective of the mathematical theory of principal fiber bundles, our definition of manifold convolution is entirely natural. Indeed it is clear that gauge invariance is not just nice to have, but necessary in order for the convolution to be geometrically well-defined.

In future work, we plan to implement gauge CNNs on general manifolds and work on further scaling of spherical CNNs. Our chart-based approach to manifold convolution should in principle scale to very large problems, thus opening the door to learning from high-resolution planetary scale spherical signals that arise in the earth and climate sciences.

---

Table 2. Climate pattern segmentation accuracy (%) for BG, TC and AR classes plus mean accuracy and average precision (mAP).

<table>
<thead>
<tr>
<th>Model</th>
<th>BG</th>
<th>TC</th>
<th>AR</th>
<th>Mean</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mudigonda et al.</td>
<td>97</td>
<td>74</td>
<td>65</td>
<td>78.67</td>
<td>-</td>
</tr>
<tr>
<td>Jiang et al.</td>
<td>97</td>
<td>94</td>
<td>93</td>
<td>94.67</td>
<td>-</td>
</tr>
<tr>
<td>Ours (S2R)</td>
<td>97.3</td>
<td>97.8</td>
<td>97.3</td>
<td>97.5</td>
<td>0.686</td>
</tr>
<tr>
<td>Ours (R2R)</td>
<td>97.4</td>
<td>97.9</td>
<td>97.8</td>
<td>97.7</td>
<td>0.759</td>
</tr>
</tbody>
</table>
Acknowledgements

We would like to thank Chiyu “Max” Jiang and Mayur Mudigonda for help obtaining and interpreting the climate data, and Erik Verlinde for helpful discussions.

References


Schuller, F. Lectures on the Geometrical Anatomy of Theoretical Physics, 2016.


Steenrod, N. The Topology of Fibre Bundles. 1951.


Supplementary Material

7. Recommended reading

For more information on manifolds, fiber bundles, connections, parallel transport, the exponential map, etc., we highly recommend the lectures by Schuller (2016), as well as the book Nakahara (2003) which explain these concepts very clearly and at a useful level of abstraction.

For further study, we recommend (Sharpe, 1997; Shoshichi Kobayashi, 1963; Husemöller, 1994; Steenrod, 1951; Wendl, 2008; Crane, 2014).

8. Mathematical Theory & Physics Analogy

From the perspective of the theory of principal fiber bundles, our work can be understood as follows. A fiber bundle $E$ is a space consisting of a base space $B$ (the manifold $M$ in our paper), with at each point $p \in B$ a space $E_p$ called the fiber at $p$. The bundle is defined in terms of a projection map $\pi : E \to B$, which determines the fibers as $E_p = \pi^{-1}(p)$. A principal bundle is a fiber bundle where the fiber $F$ carries a transitive and free right action of a group $G$ (the structure group).

One can think of the fiber $E_p$ of a principal bundle as a generalized space of frames at $p$. Due to the free and transitive action of $G$ on $E_p$, we have that $E_p$ is isomorphic to $G$ as a $G$-space, meaning that it looks like $G$ except that it does not have a distinguished origin or identity element as $G$ does (i.e. there is no natural choice of frame).

A gauge transformation is then defined as a principal bundle automorphism, i.e. a map from $P \to P$ that maps fibers to fibers in a $G$-equivariant manner. Sometimes the automorphism is required to fix the base space, i.e. project down to the identity map via $\pi$. Such a $B$-automorphism will map each fiber onto itself, so it restricts to a $G$-space automorphism on each fiber.

Given a principal bundle $P$ and a vector space $V$ with representation $\rho$ of $G$, we can construct the associated bundle $P \times_\rho V$, whose elements are the equivalence classes of the following equivalence relation on $P \times V$:

$$ (p, v) \sim (pg, \rho(g^{-1})v). $$

The associated bundle is a fiber bundle over the same base space as $P$, with fiber isomorphic to $V$.

A (matter) field is described as a section of the associated bundle $A$, i.e. a map $\sigma : B \to A$ that satisfies $\pi \circ \sigma = 1_B$. Locally, one can describe a section as a function $B \to V$ (as we do in the paper), but globally this is not possible unless the bundle is trivial.

The group of automorphisms of $P$ (gauge transformations) acts on the space of fields (sections of the associated bundle). It is this group that we wish to be equivariant to.

From this mathematical perspective, our work amounts to replacing the principal $G$ bundle $H \to G/H$ used in the work on regular and steerable G-CNNs of Cohen et al. (2018a,c), by another principal $G$ bundle, namely the frame bundle of $M$. Hence, this general theory can describe in a unified way the most prominent and geometrically natural methods of geometrical deep learning (Masci et al., 2015; Boscaini et al., 2016), as well as all G-CNNs on homogeneous spaces.

Indeed, if we build a gauge equivariant CNN on a homogeneous space $H/G$ (e.g. the sphere $S^2 = SO(3)/SO(2)$), it will (under mild conditions) automatically be equivariant to the left action of $H$ also. To see this, note that the left action of $H$ on itself (the total space of the principal $G$ bundle) can be decomposed into an action on the base space $H/G$ (permuting the fibers), and an action on the fibers (cosets) that factors through $G$ (see e.g. Sec. 2.1 of (Cohen et al., 2018c)). The action on the base space preserves the local neighbourhoods from which we compute filter responses, and equivariance to the action of $G$ is ensured by the kernel constraint. Since G-CNNs (Cohen et al., 2018a) and gauge equivariant CNNs employ the most general equivariant map, we conclude that they are indeed the same, for bundles $H \to H/G$. Thus, “gauge theory is all you need”. (We plan to expand this argument in a future paper)

Most modern theories of physics are gauge theories, meaning they are based on this mathematical framework. In such theories, any construction is required to be gauge invariant (i.e. the coefficients must be gauge equivariant), for otherwise the predictions will depend on the way in which we choose to represent physical quantities. This logic applies not just to physics theories, but, as we have argued in the paper, also to neural networks and other models used in machine learning. Hence, it is only natural that the same mathematical framework is applicable in both fields.

9. Deriving the kernel constraint

The gauge equivariant convolution is given by

$$ (K * f)(p) = \int_{B^d} K(q)\rho_m(g_{p-q})f(q)dv. $$

Under a gauge transformation, we have:

$$ v \mapsto g^{-1}_p v, \quad f(q) \mapsto \rho_m(g^{-1}_p) f(q), $$

$$ w_p \mapsto w_p g_p, \quad g_{p-q} \mapsto g^{-1}_p g_{p-q} g_q. $$

It is more common to use the letter $G$ for the supergroup and $H$ for the subgroup, but that leads to a principal $H$-bundle $G \to G/H$, which is inconsistent with the main text, where we use a principal $G$ bundle. So we swap $H$ and $G$ here.
Gauge Equivariant CNNs

It follows that \( q_v \) is unchanged, because \( q_v = \exp_p w_p v \mapsto \exp_p (w_g g_p) (g_p^{-1} v) = q_v \). Substituting the rest in the convolution equation, we find

\[
\int_{\mathbb{S}^2} K(g_p^{-1} v) \rho_{\text{in}}(g_p^{-1} g_r v, g_q v) \rho_{\text{in}}(g_q^{-1}) f(q_v) dv = \int_{\mathbb{S}^2} K(g_p^{-1} v) \rho_{\text{in}}(g_p^{-1}) f(q_v) dv
\]

Now if \( K(g_p^{-1} v) = \rho_{\text{out}}(g_p^{-1}) K(v) \rho_{\text{in}}(g_p) \) (i.e. \( K \) satisfies the kernel constraint), then we get

\[
(K \ast f)(p) = \rho_{\text{out}}(g_p^{-1}) (K \ast f)(p),
\]

i.e. \( K \ast f \) transforms as a \( \rho_{\text{out}} \)-field under gauge transformations.

10. Additional information on experiments
10.1. MNIST experiments

Our main model consists of 7 convolution layers and 3 linear layers. The first layer is a scalar-to-regular gauge equivariant convolution layer, and the following 6 layers are regular-to-regular layers. These layers have 8, 16, 16, 24, 24, 32, 64 output channels, and stride 1, 2, 1, 2, 1, 2, 1, respectively.

In between convolution layers, we use batch normalization (Ioffe & Szegedy, 2015) and ReLU nonlinearities. When using batch normalization, we average over groups of 6 feature maps, to make sure the operation is equivariant. Any pointwise nonlinearity (like ReLU) is equivariant, because we use only trivial and regular representations realized by permutation matrices.

After the convolution layers, we perform global pooling over spatial and orientation channels, yielding an invariant representation. We map these through 3 FC layers (with 64, 32, 10 channels) before applying softmax.

The other models are obtained from this one by replacing the convolution layers by scalar-to-regular + orientation pooling (S2R) or scalar-to-scalar (S2S) layers, or by disabling G-padding (NP) and/or kernel expansion (NE), always adjusting the number of channels to keep the number of parameters roughly the same.

The Spherical CNN (S2CNN) is obtained from the R2R model by replacing the S2R and R2R layers by spherical and SO(3) convolution layers, respectively, keeping the number of channels and strides the same. The Spherical CNN uses a different grid than the Icosahedral CNN, so we adapt the resolution / bandwidth parameter \( B \) to roughly match the resolution of the Icosahedral CNN. We use \( B = 26 \), to get a spherical grid of size \( 2B \times 2B = 52 \times 52 \). Note that this grid has higher resolution at the poles, and lower resolution near the equator, which explains why the S2CNN performs a bit worse when trained on rotated data instead of digits projected onto the north-pole. To implement strides, we reduce the output bandwidth by 2 at each layer with stride.

The spherical convolution takes a scalar signal on the sphere as input, and outputs scalar signals on SO(3), which is analogous to a regular field over the sphere. SO(3) convolutions are analogous to R2R layers. We note that this is a stronger Spherical CNN architecture than the one used by (Cohen et al., 2018b), which achieves only 96% accuracy on spherical MNIST.

The models were trained for 60 epochs, or 1 epoch of the \( 60 \times \) augmented dataset (where each instance is transformed by each icosahedron symmetry \( g \in \mathcal{I} \), or by a random rotation \( g \in \text{SO}(3) \)).

10.2. Climate experiments

For the climate experiments, we used a U-net with regular-to-regular convolutions. The first layer is a scalar-to-regular convolution with 16 output channels. The downsampling path consists of 5 regular-to-regular layers with stride 2, and 32, 64, 128, 256, 256 output channels. The downsampling path takes as input a signal with resolution \( r = 5 \) (i.e. 10242 pixels), and outputs one at \( r = 0 \) (i.e. 12 pixels).

The decoder is the reverse of the encoder in terms of resolution and number of channels. Upsampling is performed by bilinear interpolation (which is exactly equivariant), before each convolution layer (which uses stride 1). As usual in the U-net architecture, each layer in the upsampling path takes as input the output of the previous layer, as well as the output of the encoder path at the same resolution.

Each convolution layer is followed by equivariant batchnorm and ReLU.

The model was trained for 15 epochs with batchsize 15.

10.3. 2D-3D-S experiments

For the 2D-3D-S experiments, we used a residual U-Net with the following architecture.

The input layer is a scalar-to-regular layer with 8 channels, followed by batchnorm and relu. Then we apply 4 residual blocks with 16, 32, 64, 64 output channels, each of which uses stride=2. In the upsampling stream, we use 32, 16, 8, 8 channels, for the residual blocks, respectively. Each upsampling layer receives input from the corresponding downsampling layer, as well as the previous layer. Upsampling is performed using bilinear interpolation, and downsampling by hexagonal max pooling.

The input resolution is \( r = 5 \), which is downsampled to \( r = 1 \) by the downsampling stream.

Each residual block consists of a convolution, batchnorm, skipconnection, and ReLU.
11. Computational complexity analysis of Spherical and Icosahedral CNNs

One of the primary motivations for the development of the Icosahedral CNN is that it is faster and more scalable than Spherical CNNs as originally proposed. The Spherical CNN as implemented by (Cohen et al., 2018b) uses feature maps on the sphere $S^2$ and rotation group $SO(3)$ (the latter of which can be thought of a regular field on the sphere), sampled on the SOFT grids defined by (Kostelec & Rockmore, 2007), which have shape $2B \times 2B$ and $2B \times 2B \times 2B$, respectively (here $B$ is the bandwidth / resolution parameter). Specifically, the grid points are:

$$
\alpha_{j_1} = \frac{2\pi j_1}{2B}, \\
\beta_k = \frac{\pi(2k+1)}{4B}, \\
\gamma_{j_2} = \frac{2\pi j_2}{2B},
$$

where $(\alpha_{j_1}, \beta_k)$ form a spherical grid and $(\alpha_{j_1}, \beta_k, \gamma_{j_2})$ form an SO(3) grid (for $j_1, k, j_2 = 0, \ldots 2B - 1$). These grids have two downsides.

Firstly, because the SOFT grid consists of equal-latitude rings with a fixed number of points (2B), the spatial density of points is inhomogeneous, with a higher concentration of points near the poles. To get a sufficiently high sampling near the equator, we are forced to oversample the poles, and thus waste computational resources. For almost all applications, a more homogeneous grid is more suitable.

The second downside of the SOFT grid on SO(3) is that the spatial resolution ($2B \times 2B; \alpha, \beta$) and angular resolution ($2B; \gamma$) are both coupled to the same resolution / bandwidth parameter $B$. Thus, as we increase the resolution of the spherical image, the number of rotations applied to each filter is increased as well, which is undesirable.

The grid used in the Icosahedral CNN addresses both concerns. It is spatially very homogeneous, and we apply the filters in 6 orientations, regardless of spatial resolution.

The generalized FFT algorithm used by (Cohen et al., 2018a) only works on the SOFT grid. Generalized FFTs for other grids exist (Kunis & Potts, 2003), but are harder to implement. Moreover, although the (generalized) FFT can improve the asymptotic complexity of convolution for large input signals, the FFT-based convolution actually has worse complexity if we assume a fixed filter size. That is, the SO(3) convolution (used in most layers of a typical Spherical CNN) has complexity $O(B^3 \log B)$ which compares favorably to the naive $O(B^3)$ spatial implementation. However, if we use filters with a fixed (and usually small) size, the complexity of a naive spatial implementation reduces to $O(B^3)$, which is slightly better than the FFT-based implementation. Furthermore, because the Icosahedral CNN uses a fixed number of orientations per filter (i.e. 6), its computational complexity is even better: it is linear in the number of pixels of the grid, and so comparable to $O(B^2)$ for the SOFT grid.

The difference in complexity is clearly visible in Figures 7 and 8, below. On the horizontal axis, we show the grid resolution $r$ for the icosahedral grid $H_r$ (for the spherical CNN, we use a SOFT grid with roughly the same number of spatial points). On the vertical axis, we show the amount of wallclock time (averaged over 100 runs) and memory required to run an SO(3) convolution (S2CNN) or a regular-to-regular gauge equivariant convolution (IcoNet) at that resolution. Note that since the number of grid points is exponential in $r$, and we use a logarithmic vertical axis, the figures can be considered log-log plots. Both plots were generated by running a single regular to regular convolution layer at the corresponding resolution $r$ with 12 input and output channels. For a fair comparison with IcoCNNs we chose filter grid parameters so3.near_identity_grid(n.alpha=6, max_beta=np.pi/16, n.beta=1, max_gamma=2*np.pi, n_gamma=6) for the spherical convolution layer. To guarantee a full GPU utilization, results were measured on an as large as possible batch size per datapoint and subsequently normalized by that batch size.

![Figure 7. Comparison of computational cost (in wallclock time) of Icosahedral CNNs (IcoNet) and Spherical CNNs (S2CNN, (Cohen et al., 2018b)), at increasing grid resolution $r$.](image)

As can be seen in Figure 7, the computational cost of running the S2CNN dramatically exceeds the cost of running the IcoCNN, particularly at higher resolutions. We did not run the spherical CNN beyond resolution $r = 6$, because the network would not fit in GPU memory even when using batch size 1.

As shown in Figure 8, the Spherical CNN at resolution $r = 6$ uses about 10GB of memory, whereas the Icosahedral CNN
A final note on scalability. For some problems, such as the analysis of high resolution global climate or weather data, it is unlikely that even a single feature map will fit in memory at once on current or near-term future hardware. Hence, it may be useful to split large feature maps into local charts, and process each one on a separate compute node. For the final results to be globally consistent (so that each compute node makes equivalent predictions for points in the overlap of charts), gauge equivariance is indispensable.

12. Details on G-Padding

In a conventional CNN, one has to pad the input feature map in order to compute an output of the same size. Although the icosahedron itself does not have a boundary, the charts do, and hence require padding before convolution. However, in order to faithfully simulate convolution on the icosahedron via convolution in the charts, the padding values need to be copied from another chart instead of e.g. padding by zeros. In doing so, a gauge transformation may be required.

To see why, note that the conv2d operation, which we use to perform the convolution in the charts, implicitly assumes that the signal is expressed relative to a fixed global gauge in the plane, namely the frame defined by the x and y axes. This is because the filters are shifted along the x and y directions by conv2d, and as they are shifted they are not rotated. So the meaning of “right” and “up” doesn’t change as we move over the plane; the local gauge at each position is aligned with the global x and y axes.

Hence, it is this global gauge that we must use inside the charts shown in Figure 4 (right) of the main paper. It is important to note that although all frames have the same numerical expression $e_1 = (1,0), e_2 = (0,1)$ relative to the x and y axes, the corresponding frames on the icosahedron itself are different for different charts. Since feature vectors are represented by coefficients that have a meaning only relative to a frame, they have a different numerical expression in different charts in which they are contained. The numerical representations of a feature vector in two charts are related by a gauge transformation.

To better understand the gauge transformation intuitively, consider a pixel $p$ on a colored edge in Fig. 4 of the main paper, that lies in multiple charts. Now consider a vector attached at this pixel (i.e. in $T_p M$), pointing along the colored edge. Since the colored edge may have different orientations when pictured in different charts, the vector (which is aligned with this edge) will also point in different directions in the charts, when the charts are placed on the plane together as in Figure 4. More specifically, for the choice of charts we have made, the difference in orientation is always one “click”, i.e. a rotation by plus or minus $2\pi/6$. This is the gauge transformation $g_{ij}(p)$, which describes the transformation at $p$ when switching between chart $i$ and $j$.

The transformation $g_{ij}(p)$ acts on the feature vector at $p$ via the matrix $\rho(g_{ij}(p))$, where $\rho$ is the representation of $G = C_6$ associated with the feature space under consideration. In this work we only consider two kinds of representations: scalar features with $\rho(g) = 1$, and regular features with $\rho$ equal to the regular representation:

$$\rho(2\pi/6) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (14)$$

That is, a cyclic permutation of 6 elements. Since $2\pi/6$ is a generator of $C_6$, the value of $\rho$ at the other group elements is...
determined by this matrix: $\rho(k \cdot 2\pi/6) = \rho(2\pi/6)^k$. If the feature vector consists of multiple scalar or regular features, we would have a block-diagonal matrix $\rho(g_{ij}(p))$.

We implement G-padding by indexing operations on the feature maps. For each position $p$ to be padded, we pre-compute $g_{ij}(p)$, which can be $+1 \cdot 2\pi/6$ or $0$ or $-1 \cdot 2\pi/6$. We use these to precompute four indexing operations (for the top, bottom, left and right side of the charts).