Convolution-based nonlinear functions on tensor images

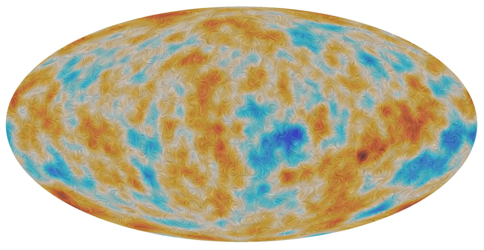
Wilson Gregory, David W. Hogg, Ben Blum-Smith,
Maria Teresa Arias, Kaze W. K. Wong, Soledad Villar

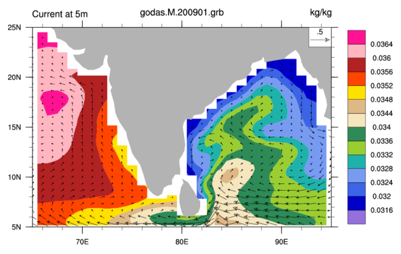
Abstract: Convolutional neural networks and their ilk have been wildly successful for a plethora of learning tasks involving images for more than a decade. These methods assume that the input is a scalar image representing the intensity in each pixel, possibly in multiple channels for color images. In natural-science domains however, image-like data sets might have vectors (velocity, say), tensors (polarization, say), pseudovectors (magnetic field, say), or other geometric objects in each pixel, and treating the components of these objects as independent channels in a CNN neglects their structure entirely. Our formulation, the *GeometricImage-Net*, combines a geometric generalization of convolution with products, tensor index contractions, and tensor index permutations to construct functions of geometric images that put the tensor structure to work. The framework permits, with a very simple adjustment, restriction to function spaces that are exactly equivariant to translations, discrete rotations, and reflections, and our results suggest that we are able to express all such equivariant functions from geometric images to geometric images. In our numerical experiments we see the power of the equivariant GeometricImage-Net in its ability to generalize well to unseen data in a small training data set regime.

1 Introduction

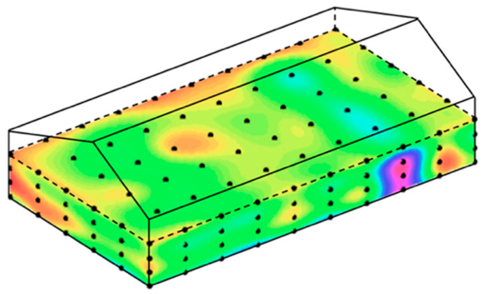
Contemporary natural science and engineering is replete with data sets that are images, lattices, or grids of geometric objects. These might be observations of intensities (scalars), magnetic fields (pseudovectors), or polarizations (2-tensors) on a surface or in a volume. They might be the inputs or outputs of a simulation where the initial conditions or fields are specified on a regular grid. See Figure 1 for some examples. Any lattice of vectors or tensors can be seen as a generalization of the concept of an image in which the intensity in each pixel is replaced with a geometric object (scalar, vector, pseudovector, tensor). These objects are *geometric* in the sense that they are defined in terms of their transformation properties under geometric operators such as rotation, translation, and reflection. Thus there is a need for machine learning methods designed for *geometric images*—lattices or grids of scalars, vectors, and tensors. There are already countless applications of machine learning in contexts in which the input data are geometric images, including examples in essentially all natural-science disciplines.

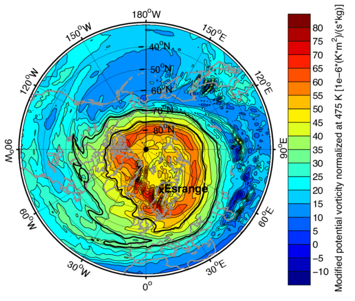
At the present day, the go-to tools for machine learning with images are convolutional neural networks (CNNs; [31]) and their many descendants, including residual networks (ResNets) [25], dense networks (DenseNets)[26], and attention mechanisms such as transformers [41]. Other recent tools for machine learning with images include generative adversarial networks (GANs)[19] for image synthesis and style transfer, and recurrent neural networks (RNNs)[38] for tasks such as image captioning and video analysis. Additionally, transfer learning [52] has emerged as a powerful technique for leveraging pre-trained models on large image datasets to

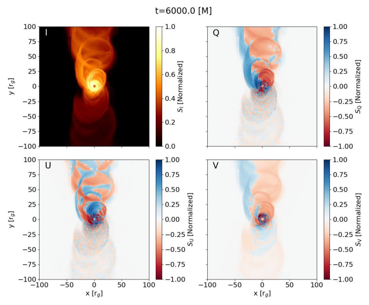

(a) temperature and polarization



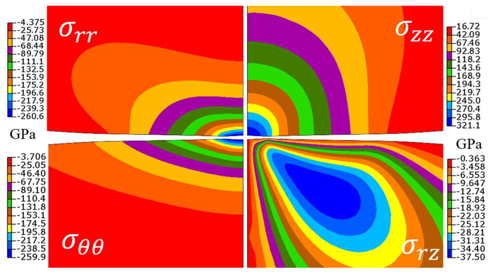
(b) salinity and current


(c) temperature


Modified potential vorticity normalized at 475 K [1e-6*(K*m²)/(s*kg)]
Esrange
(d) vorticity



(e) intensity and polarization



(f) stress tensor

Figure 1: Examples of geometric images in the natural sciences. *(a)* A visualization of a temperature map and a polarization map from the ESA *Planck* Mission [9]. The color map shows a temperature field (a scalar or $0_{(+)}$-tensor) on the sphere, and the whiskers show the principal eigenvector direction of a $2_{(+)}$-tensor field in two dimensions. In detail the underlying data are represented on a pixel grid (healpixel [22]) on the sky (a 2-sphere). *(b)* Two-dimensional maps of ocean current (shown with arrows; a vector or $1_{(+)}$-tensor field) and ocean salinity (shown with color; a scalar or $0_{(+)}$-tensor field) at a depth of 5 m [1]. *(c)* A three-dimensional map of temperature (a scalar or $0_{(+)}$-tensor field) based on sensors distributed throughout the volume of a granary [44]. *(d)* A two-dimensional map of potential vorticity (a pseudoscalar or $0_{(-)}$-tensor field) in the Earth's atmosphere, measured for the purposes of predicting storms [33]. *(e)* Two-dimensional maps on the sky of intensity I and the three independent components Q, U, V of the electromagnetic polarization $2_{(+)}$-tensor, from a simulation of a jet outflow from an accreting black hole [Davelaar et al, in preparation]. *(f)* Components of the three-dimensional stress tensor (a $2_{(+)}$-tensor field) in a diamond anvil cell, which is used to study the behavior of samples at exceedingly high pressures [32].

improve performance on smaller or specialized datasets.

Traditional CNNs are designed to work on one- or few-channel images in which the early layers of the network involve image convolutions with learned filters followed by the application of pointwise nonlinearities. In typical contexts, the channels of multi-channel input images will be something like the red, green, and blue channels of a color image; these can be combined arbitrarily in the layers of the CNN. When these CNN-based tools are applied to lattices of vectors for example, typically the components of the vectors are treated just as channels of the input image and then everything proceeds as with multi-channel color images. This is not good!

Actually, it *is* good: There are many successful projects that have repurposed CNNs in this way to work on geometric images. But there are even better choices. Here we propose a set of tools that generalize the concept of convolution to apply to geometric images such that the outputs of the convolutions are also geometric images, obeying the same geometric transformation rules as the inputs.

The fundamental observation inspiring this work is that when an arbitrary function is applied to the components of vectors and tensors, the geometric structure of these objects is destroyed. There are strict rules, dating back to the early days of differential geometry [37], about how geometric objects can be combined to produce new geometric objects, consistent with coordinate freedom and transformation rules. These rules constitute a theme of [39], where they are combined into a *geometric principle*. In previous work [42, 43, 50] we have capitalized on the geometric principle to develop modified machine-learning methods that are restricted to exactly obey group-theoretic equivariances in physics contexts. Here we use these rules to create a comprehensive set of tools that parameterize functions that take geometric images as input and produce geometric images as output.

Tensors can be defined—and distinguished from mere arrays of numbers—in two ways. In one, a tensor of order k is a k-multilinear function of k vector inputs that returns a scalar, which is an object with a value that is invariant to changes in the coordinate system ([39] Section 1.3). In the other, a tensor of order k is defined by the way that its components transform under rotations ([39] Section 1.6). We will take the latter point of view, and this definition will be made precise in Section 2.1.

There are two ways to think about transformations—*alias* and *alibi*. In the former (alias), the idea is that the transformation is applied to the coordinate system, not the vectors and tensors themselves. This transformation leaves the geometric objects unchanged but all of their components change because they are now being represented in a changed coordinate system. In the latter (alibi), the idea is that the coordinate system is fixed and all of the geometric objects are taken through an identical transformation. In either case—alias or alibi—the key idea is that *all* of the components of *all* of the vectors and tensors in play must be changed correspondingly, and at the same time. The geometric principle requires that for any function, all the inputs, constants, parameters, and outputs must undergo the same coordinate transformations simultaneously. In other words, all valid functions will be fundamentally equivariant with respect to coordinate transformations.

We are motivated in this work to help solve problems in the natural sciences and engineering, where geometric images abound. However, we conjecture that these tools are probably very useful even for standard machine-learning image-recognition and image-regression tasks. After all, even standard images are measurements of a scalar, the intensity of light, at a regular grid of points on a two-dimensional surface. The laws of physics still govern the objects in a photograph and how light travels from the objects to the camera, so we may still expect to benefit from the rules of geometry.

These rules of geometry—the consequences of the geometric principle—are roughly as follows: A k-tensor object (tensor of order k) in d dimensions has k indices, each of which can take a value from 1 to d; that is, the k-tensor is an element of $(\mathbb{R}^d)^{\otimes k}$. It can be contracted to a $(k-2)$-tensor object by identifying a pair of indices and summing over them. A k-tensor and a k'-tensor can be multiplied (outer product) to make a $(k+k')$-tensor object, and then contractions can bring down the order. 1-tensor objects are called vectors and 0-tensor objects are called scalars. There are also negative-parity versions of all these (pseudovectors, pseudoscalars, and pseudotensors), and parity-changing contractions using the Levi-Civita symbol, so in what follows we will define $k_{(p)}$-tensors that have k indices and a parity $p \in \{-1, +1\}$ (sometimes denoted "$-$" and "$+$" below). Two objects can only be added or subtracted if they have the same order k and parity p. These rules define objects that can be given transformation rules under rotation and reflection such that functions made of these operations are coordinate free, or equivariant to any change of coordinate system.

The symmetries that suggest these rules are continuous symmetries. But of course images are usually—and for our purposes—discrete grids of values. This suggests that in addition to the continuous symmetries respected by the tensor objects in the image pixels there will be discrete symmetries for each geometric image taken as a whole. We will define these discrete symmetry groups and use them to define a useful kind of group equivariance for functions of geometric images. This equivariance, it turns out, is very easy to enforce, even for nonlinear functions of geometric images, provided that we compose our nonlinear functions from simple geometric operations, including our geometric generalization of convolution. When we enforce this equivariance, the convolution filters that appear look very much like the differential operators that appear in discretizations of vector calculus.

Our contribution: The rest of the paper is organized in the following manner. Section 2 defines geometric objects, geometric images, and the operations on each. Section 3 discusses equivariance of functions of geometric images with some important results building off of [29] and [7]. Section 4 describes how to explicitly count these equivariant functions using a result of Molien from 1897. Sections 5 and 6 describe how to build a GeometricImage-Net and present a couple of small problems with numerical results. Finally, Section 7 discusses related work. The majority of the supporting propositions and proofs have been sequestered to the Appendix, as has a larger exploration of related work.

2 Geometric Objects and Geometric Images

We define the geometric objects and geometric images that we use to generalize classical images in scientific contexts in Section 2.1 and Section 2.2. The main point is that the channels of geometric images—which will be like the components of vectors and tensors—are not independent. There is a set of allowed operations on geometric objects that respect the structure and the coordinate freedom of these objects.

2.1 Geometric objects

The geometric principle implies that geometric objects should be coordinate-free scalars, vectors, and tensors, or their negative-parity pseudo counterparts. To define these objects we start by stating the coordinate transformations, which for now will be given by the orthogonal group.

We fix d, the dimension of the space, which will typically be 2 or 3. The geometric objects are vectors and tensors. The orthogonal group $O(d)$ is the space of isometries of $\mathbb{R}^d$ that fix the origin. It acts on vectors and pseudovectors $v \in \mathbb{R}^d$ in the following way:

$$g \cdot v = \det(M(g))^{\frac{1-p}{2}}\, M(g)\, v \tag{1}$$

where for $g \in O(d)$, $M(g) \in \mathbb{R}^{d\times d}$ is the standard matrix representation of g (i.e. $M(g)^\top M(g) = I$) and $p \in \{-1, +1\}$ is the parity of v. If $p = +1$ we obtain the standard $O(d)$ action on $\mathbb{R}^d$ *vectors*. If $p = -1$ we obtain the $O(d)$ action on what in physics are known as *pseudovectors*.

The objects are defined by the actions that they carry in the following sense: if F is a function with geometric inputs, outputs, and parameters, then F must be coordinate-free. In other words $F(g \cdot v) = g \cdot F(v)$ for all v and all g. This is the mathematical concept of equivariance which we will explore further in Section 3.

Definition 1 ($k_{(p)}$-tensors)**.** The space $\mathbb{R}^d$ equipped with the action $O(d)$ defined by (1) is the space of $1_{(p)}$*-tensors*. If v_i is a $1_{(p_i)}$-tensor, then $T := v_1 \otimes \ldots \otimes v_k$ is a *rank-1* $k_{(p)}$*-tensor*, where $p = \prod_{i=1}^k p_i$ and the action of $O(d)$ is defined as

$$g \cdot (v_1 \otimes \ldots \otimes v_k) = (g \cdot v_1) \otimes \ldots \otimes (g \cdot v_k)\ . \tag{2}$$

Higher rank $k_{(p)}$-tensors are defined as linear combinations of rank-1 $k_{(p)}$-tensors where the action of $O(d)$ is extended linearly. The set of $k_{(p)}$-tensors in d dimensions is denoted $\mathcal{T}_{d,k,p}$.

Remark (terminology and notation)**.** The parity p is a signed bit, either $+1$ for positive parity or -1 for negative parity. Note the distinction between the *order* k of the $k_{(p)}$-tensor, and the rank of the tensor. We could have a $2_{(+)}$-tensors of rank 1, like those we use in Definition 1.

Remark (universality of transformations)**.** Critically, when a transformation is applied to any $k_{(p)}$-tensor, it must be applied to every other geometric object—every scalar, vector, and tensor of both parities—involved in any relevant mathematical expression. This includes all constants, and all inputs and outputs to any scalar, vector, or tensor functions. Related to this, there are both alias and alibi points of view that can be taken towards (2); that is, it can be seen as defining a change made to every tensor in a fixed coordinate system, or it can be seen as a change to the coordinate system in which every tensor is represented.

In physics the $1_{(+)}$-tensors (such as velocities) are known as *vectors*, the $1_{(-)}$-tensors (such as angular momenta) are known as *pseudovectors*, the $0_{(+)}$-tensors (such as rest masses) are known as *scalars*, the $0_{(-)}$-tensors (such as surface vorticities) are known as *pseudoscalars*, the $k_{(-)}$-tensors with $k \geq 2$ are known as *pseudotensors*, and finally the $k_{(+)}$-tensors with $k \geq 2$ are the things that are commonly known as *tensors*. In general, any $k_{(p)}$-tensor can be written as a sum of outer products of order-1 ($k = 1$) tensors (vectors and pseudovectors), where each term in the sum is an outer product of k order-1 tensors and the parity p is the product of the parities of the input order-1 tensors.

Definition 2 (outer products of tensors)**.** Given $a \in \mathcal{T}_{d,k,p}$ and $b \in \mathcal{T}_{d,k',p'}$, the *outer product*, denoted $a \otimes b$, is a tensor in $\mathcal{T}_{d,k+k',p\,p'}$ defined as $[a \otimes b]_{i_1,\ldots,i_{k+k'}} = [a]_{i_1,\ldots,i_k}\,[b]_{i_{k+1},\ldots,i_{k+k'}}$.

Definition 3 (Einstein summation notation)**.** We use *Einstein summation notation* where outer products are written in component form, and repeated indices are summed over. For example, in this notation, the product of two $2_{(+)}$-tensors (represented as two $d \times d$ matrices A and B) is written as

$$[A\,B]_{i,j} = [A]_{i,k}\,[B]_{k,j} := \sum_{k=1}^{d} [A]_{i,k} [B]_{k,j} \tag{3}$$

where $[A]_{i,k}$ is the i, k element of matrix A, and the sum from 1 to d on repeated index k is implicit in the middle expression. This notation works for tensor expressions of any order, provided that every index appears either exactly once, so it isn't summed over, or exactly twice, so it is summed over.

Remark (lower and upper indices)**.** In the original Einstein summation notation [14], or Ricci calculus [37], a distinction is made between lower and upper indices, which correspond to covariant and contravariant components. The pairs of indices that are summed always have one member of the pair an upper index and one member a lower index. We drop the upper/lower distinction here because we will work with intrinsically flat images that implicitly have the Riemmannian metric given by the identity matrix, such that there is no numerical difference between covariant and contravariant component values for a given object. That said, there truly is a distinction (for example, if a spatial displacement is a contravariant vector, the gradient of a scalar function with respect to that spatial displacement is a covariant vector), so there might be advantages to reinstating this distinction. HOGG: I don't understand the distinction example here -Wilson

In summation notation, the group action of (1) on $k_{(p)}$-tensor b is explicitly written

$$[g \cdot b]_{i_1,\ldots,i_k} = \det(M(g))^{\frac{1-p}{2}}\, [b]_{j_1,\ldots,j_k}\, [M(g)]_{i_1,j_1} \cdots [M(g)]_{i_k,j_k} \tag{4}$$

for all $g \in O(d)$, where $[b]_{i_1,\ldots,i_k} \in \mathbb{R}$ is a component of b, $[M(g)]_{i,j} \in \mathbb{R}$ is the i, j element of the matrix representation of g, and all the i_m and j_m are indices in the range $1, \ldots, d$. For example, a $2_{(+)}$-tensor has the transformation property $[g \cdot b]_{i,j} = [b]_{k,\ell}\,[M(g)]_{i,k}\,[M(g)]_{j,\ell}$, which, in normal matrix notation, is written as $g \cdot b = M(g)\, b\, M(g)^\top$.

We consider two special tensors that will be important for the definition of our models, the Kronecker delta and the Levi-Civita symbol.

Definition 4 (Kronecker delta)**.** The *Kronecker delta* δ is a $2_{(+)}$-tensor represented by the identity matrix, namely the object with two indices ij such that it has the value $+1$ when the two indices have the same value ($i = j$), and 0 otherwise.

Definition 5 (Levi-Civita symbol)**.** The *Levi-Civita symbol* in dimension $d \geq 2$ is a $d_{(-)}$-tensor ϵ such that if the d indices are not repeated and in an even-permutation order the value is $+1$ and if the d indices are not repeated and in an odd-permutation order the value is -1, and it has the value 0 in all other cases.

Definition 6 (contractions)**.** Given tensor $a \in \mathcal{T}_{d,k,p}$, where $k \geq 2$, and given $\mu, \nu \in [k], \mu \neq \nu$, the *contraction* $T(a, \mu, \nu) \in \mathcal{T}_{d,k-2,p}$ is defined as:

$$[T(a,\mu,\nu)]_{i_1,\ldots,i_k \setminus \{i_\mu,i_\nu\}} = [\delta]_{i_\mu i_\nu} [a]_{i_1,\ldots,i_\mu,\ldots,i_\nu,\ldots,i_k} \tag{5}$$

That is, we view the components of a with given fixed values for i_μ and i_ν as forming a $(k-2)_{(p)}$-tensor, and then we take the sum of these tensors of order $k-2$ where $i_\mu = i_\nu$. We can also define the composition of multiple contractions as a *multicontraction*:

$$T_M(a, (\mu_1, \mu_2), \ldots, (\mu_\ell, \mu_{\ell+1})) = T(\cdot, \mu_\ell, \mu_{\ell+1}) \circ \ldots \circ T(a, \mu_1, \mu_2)\ , \tag{6}$$

where $\mu_1, \ldots, \mu_{\ell+1} \in [k]$ are all distinct. Note that because $\mu_1, \ldots, \mu_{\ell+1}$ are integers referring to the indices of the axes being contracted, the indices may change when swapping from a multicontraction to multiple contractions. For example, if $k \geq 4$,

$$T_M(a, (1, 3), (2, 4)) = T(T(a, 1, 3), 1, 2)$$

because axes i_1 and i_3 will disappear, so i_2 becomes the new i_1 and i_4 becomes the new i_2. Finally, the *Levi-Civita contraction* is defined for $k \geq d - 1$ and $\mu_1, \ldots, \mu_{d-1} \in [k]$ distinct as the following:

$$T_{LC}(a, \mu_1, \ldots, \mu_{d-1}) = T_M(a \otimes \epsilon, (\mu_1, k + 1), \ldots, (\mu_{d-1}, k + d - 1)) \ , \tag{7}$$

where ϵ is the Levi-Civita symbol.

Remark (negative-parity objects)**.** With a slight modification of the Levi-Civita contraction, there is an invertible function that converts any negative-parity object to a positive-parity object. Thus it is possible to work without negative-parity objects at all, and we will use this idea to improve the efficiency of our algorithms for certain settings in Section 5.2. However, since negative-parity objects are important in physics and engineering (see Figure 1), we retain them in our model.

We can combine multiplication with Kronecker and Levi-Civita symbols with contractions to define relevant operations. For example the $2_{(+)}$-tensor formed by the outer product of $1_{(+)}$-tensors a and b can be contracted with the Kronecker delta to give the standard dot product $a^\top b = [a]_i \, [b]_j \, [\delta]_{ij}$, which is a $0_{(+)}$-tensor or scalar. For another example, the same $2_{(+)}$-tensor can (in $d = 3$ dimensions) be contracted with the Levi-Civita symbol to give the standard cross product $[a \times b]_k = [a]_i \, [b]_j \, [\epsilon]_{ijk}$, which is a $1_{(-)}$-tensor or pseudovector.

Definition 7 (permutations of tensor indices)**.** Given $a \in \mathcal{T}_{d,k,p}$ and permutation $\sigma \in S_k$, the *permutation of tensor indices* of a by σ, denoted a^σ, is:

$$[a^\sigma]_{i_1,\ldots,i_k} := [a]_{i_{\sigma^{-1}(1)},\ldots,i_{\sigma^{-1}(k)}} \tag{8}$$

Remark (tensors as linear functions)**.** There is an alternative definition of $k_{(p)}$-tensors in terms of geometric functions (see, for example, [39] chapter 1): A $k_{(+)}$-tensor can be thought of as representing a multilinear function of k vectors ($1_{(+)}$-tensors) that produces a scalar ($0_{(+)}$-tensor) output. For example, if A is a $4_{(+)}$-tensor, and u, v, w, x are vectors ($1_{(+)}$-tensors) then

$$\rho = [A]_{ijk\ell} \, [u]_i \, [v]_j \, [w]_k \, [x]_\ell \tag{9}$$

is a scalar ($0_{(+)}$-tensor). $k_{(-)}$-tensors can be similarly defined in terms of input vectors and an output pseudoscalar.

2.2 Geometric images and operations

We will start by considering square (or cubic or hyper-cubic) images on a d-torus. We work on a d-torus to simplify the mathematical results; all the definitions and operations will be applicable with minor adjustments to rectangular, non-toroidal arrays as well. We consider an image A in with N equally spaced pixels in each dimension for N^d pixels total. Each pixel contains a $k_{(p)}$-tensor where k and p are the same for each pixel. Let $\mathcal{T}_{d,k,p}$ be the set of $k_{(p)}$-tensors in $\mathbb{R}^d$. We define the geometric images as follows.

Definition 8 (geometric image)**.** A *geometric image* is a function $A : [N]^d \to \mathcal{T}_{d,k,p}$, where $[N] = \{0, 1, \ldots, N-1\}$. The set of geometric images is denoted $\mathcal{A}_{N,d,k,p}$. We will also consider $k_{(p)}$-tensor images on the d-torus, where $[N]^d$ is given the algebraic structure of $(\mathbb{Z}/N\mathbb{Z})^d$. The pixel index of a geometric image, often $\bar{\imath}$, is naturally a $1_{(+)}$-tensor of length d.

Definition 9 (sums of images)**.** Given $A, B \in \mathcal{A}_{N,d,k,p}$, the *sum* $A + B \in \mathcal{A}_{N,d,k,p}$ is defined as

$$(A + B)(\bar{\imath}) = A(\bar{\imath}) + B(\bar{\imath}) \tag{10}$$

for pixel $\bar{\imath}$. That is, the sums of geometric images are performed pixel-wise.

Definition 10 (scalar multiplication of images)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ and $\alpha \in \mathbb{R}$, the *scalar product* αA is defined as

$$(\alpha A)(\bar{\imath}) = \alpha A(\bar{\imath}) \ . \tag{11}$$

Similarly, we define contractions and permutations as applying an identical contraction or permutation to every pixel.

We now turn to the first major contribution of this paper, the generalization of convolution to take geometric images as inputs and return geometric images as outputs. The idea is that a geometric image of $k_{(p)}$-tensors is convolved with a geometric filter of $k'_{(p')}$-tensors to produce a geometric image that contains $(k + k')_{(p\,p')}$-tensors, where each pixel is a sum of outer products. These outer products can be contracted down to lower-order tensors using contractions (Definition 6). Note that the sidelength M of the geometric filter can be any positive odd number, but typically it will be much smaller than the sidelength N of the geometric image.

Definition 11 (geometric convolution)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ on the d-torus, and $C \in \mathcal{A}_{M,d,k',p'}$ where $M = 2m+1$ for some $m \in \mathbb{N}$, the *geometric convolution* $A * C$ is a $(k + k')_{(p\,p')}$-tensor image such that

$$(A * C)(\bar{\imath}) = \sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath} - \bar{a}) \otimes C(\bar{a} + \bar{m}) \ , \tag{12}$$

where $\bar{\imath} - \bar{a}$ is the translation of $\bar{\imath}$ by $\bar{a}$ on the d-torus pixel grid $(\mathbb{Z}/N\mathbb{Z})^d$. Additionally, $\bar{m}$ is the d length $1_{(+)}$-tensor $[m, \ldots, m]^T$. For example, if $d = 2$ and $\bar{a} = [0, 0]^T$, then $\bar{a} + \bar{m} = [m, m]^T$, the center pixel of C as we would expect.

This definition is on the torus, which we use to simplify the mathematical exposition. To define the convolution on $[N]^d$ instead of the torus, we can pad the image out with zero tensors of the corresponding order and parity. See Figure 2 for examples with a scalar and vector filter.

In addition to contractions and index permutations that act pixel-wise in geometric images, it is possible to change the image size using pooling and unpooling operations. For both pooling and unpooling, there are alternative strategies to the ones we have defined below.

Definition 12 (average pooling)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ and $b \in \mathbb{Z}^+$ such that N is divisible by b, we define $\operatorname{avg\,pool}(A, b) \in \mathcal{A}_{N/b,d,k,p}$ for pixel index $\bar{\imath}$ as:

$$\operatorname{avg\,pool}(A, b)(\bar{\imath}) = \frac{1}{b^d} \sum_{\bar{a} \in [0,b-1]^d} A(b\bar{\imath} + \bar{a}) \tag{13}$$

Definition 13 (nearest neighbor unpooling)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ and $b \in \mathbb{Z}^+$, we define $\operatorname{unpool}(A, b) \in \mathcal{A}_{Nb,d,k,p}$ for pixel index $\bar{\imath}$ as:

$$\operatorname{unpool}(A, b)(\bar{\imath}) = A(\lfloor \bar{\imath}/b \rfloor) \tag{14}$$

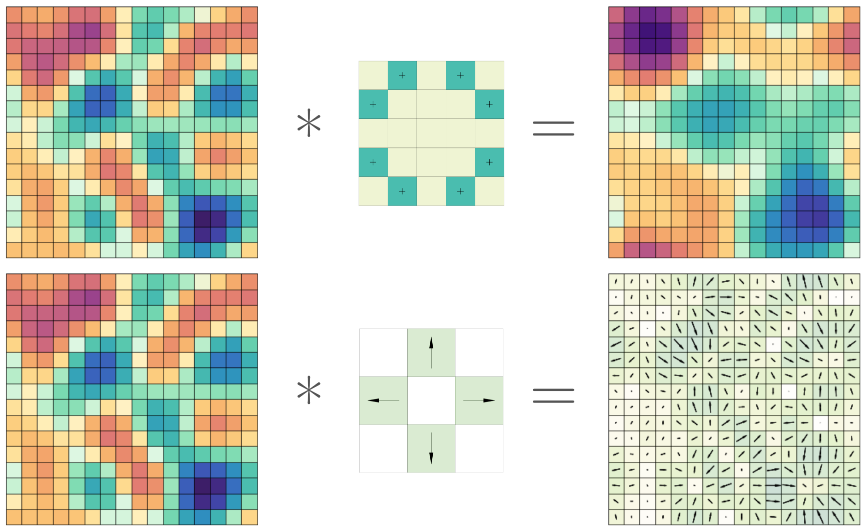


Figure 2: Convolution of a scalar image with a scalar filter and with a vector filter. Note that convolution with the vector filter results in a vector image that looks like the gradient.

where $\lfloor \bar{\imath}/b \rfloor$ denotes dividing each component of $\bar{\imath}$ by b, then taking element-wise floor operator of the resulting vector.

The convolution, contraction, index-permutation, and pooling operators above effectively span a large class of linear functions from geometric images to geometric images. Nonlinear functions can be constructed using polynomials. Polynomials in this sense will be sums of outer products of any of the linear function outputs, possibly followed by further geometric convolutions and contractions. Nonlinear functions can also be constructed by applying nonlinear functions to $0_{(+)}$-tensors (scalars), or odd nonlinear functions to $0_{(-)}$-tensors (pseudoscalars); we will return to these methods in Section 5.

Definition 14 (outer products of images)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ and $B \in \mathcal{A}_{N,d,k',p'}$, the *outer product* $A \otimes B \in \mathcal{A}_{N,d,k+k',p\,p'}$ is defined as

$$(A \otimes B)(\bar{\imath}) = A(\bar{\imath}) \otimes B(\bar{\imath}) \ . \tag{15}$$

for each pixel $\bar{\imath}$. That is, the outer products of geometric images are performed pixel-wise.

3 Functions of geometric images and equivariance

We start by defining equivariance and invariance for a general group G, and then we will describe the groups of interest and several theoretical results.

Definition 15 (Equivariance of a geometric image function)**.** Given a function on geometric images $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$, and a group G equipped with actions on $\mathcal{A}_{N,d,k,p}$ and $\mathcal{A}_{N,d,k'',p''}$, we say that f is *equivariant* to G if for all $g \in G$ and $A \in \mathcal{A}_{N,d,k,p}$ we have:

$$f(g \cdot A) = g \cdot f(A) \tag{16}$$

Likewise, f is *invariant* to G if

$$f(g \cdot A) = f(A) \ . \tag{17}$$

We may also say a geometric image is invariant to G if $g \cdot A = A$ for all $g \in G$.

Convolutional filters are widely used in machine learning for scalar images. The fundamental property of these operators are that they are translation equivariant, and that every translation equivariant linear function can be expressed as a convolution with a fixed filter, as long as the filter can be set to be as large as the image. The same property holds for geometric images.

Definition 16 (Translation of $k_{(p)}$-tensor images)**.** Given a $k_{(p)}$-tensor image A on the d-torus, and a translation $\tau \in (\mathbb{Z}/N\mathbb{Z})^d$, the action $L_\tau A$ produces a $k_{(p)}$-tensor image on the d-torus such that

$$(L_\tau A)(\bar{\imath}) = A(\bar{\imath} - \tau) \ , \tag{18}$$

where $\bar{\imath} - \tau$ is the translation of $\bar{\imath}$ by τ on the d-torus pixel grid $(\mathbb{Z}/N\mathbb{Z})^d$.

Proposition 1. A function $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k+k',p\,p'}$ is a translation equivariant linear function if and only if it is the convolution with a geometric filter $C \in \mathcal{A}_{M,d,2k+k',p'}$ followed by k contractions. When N is odd, $M = N$, otherwise $M = N + 1$.

Note that this proposition is just a special case of the general result of [29]. See appendix A for the proof.

In addition to translation symmetries, we want to consider other natural symmetries occurring in the application domains where vectors and tensors arise. Ideally we would like to apply continuous rotations to the images, but the discretized nature of images makes this challenging. For simplicity, we focus on discrete rotations, and we extend the group action to the geometric objects in these images.

Definition 17 (Group B_d of symmetries of a d-hypercube)**.** We denote by B_d the group of Euclidean symmetries of the d-dimensional hypercube.

The group B_d is often called the *hyperoctahedral group* since the d-dimensional hyperoctahedron is dual to the hypercube, so they have the same group of symmetries. The notation B_d is standard nomenclature coming from the classification theorem for finite irreducible reflection groups [27]. See Figure 3 for a depiction of the elements of B_2 acting on a vector. Because the groups B_d are subgroups of $O(d)$, all determinants of the matrix representations of the group elements are either $+1$ or -1, and the matrix representation $M(g^{-1})$ of the inverse g^{-1} of group element g is the transpose of the matrix representation $M(g)$ of group element g.

Definition 18 (Action of B_d on $k_{(p)}$-tensors)**.** Given a $k_{(p)}$-tensor b, the action of $g \in B_d$ on b, denoted $g \cdot b$, is the restriction of the action in Definition 1 to B_d which is a subgroup of $O(d)$.

Definition 19 (Action of B_d on $k_{(p)}$-tensor images)**.** Given $A \in \mathcal{A}_{N,d,k,p}$ on the d-torus and a group element $g \in B_d$, the action $g \cdot A$ produces a $k_{(p)}$-tensor image on the d-torus such that

$$(g \cdot A)(\bar{\imath}) = g \cdot A(g^{-1} \cdot \bar{\imath}) \ . \tag{19}$$

Since $\bar{\imath}$ is a $1_{(+)}$-tensor, the action $g^{-1} \cdot \bar{\imath}$ is performed by centering $\bar{\imath}$, applying the operator, then un-centering the pixel index:

$$g^{-1} \cdot \bar{\imath} = \big(M(g^{-1})(\bar{\imath} - \bar{m})\big) + \bar{m}$$

where $\bar{m}$ is the d-length $1_{(+)}$-tensor $\left[\frac{N-1}{2}, \ldots, \frac{N-1}{2}\right]^T$. If the pixel index is already centered, such as $\bar{a} \in [-m, m]^d$, then we skip the centering and un-centering.

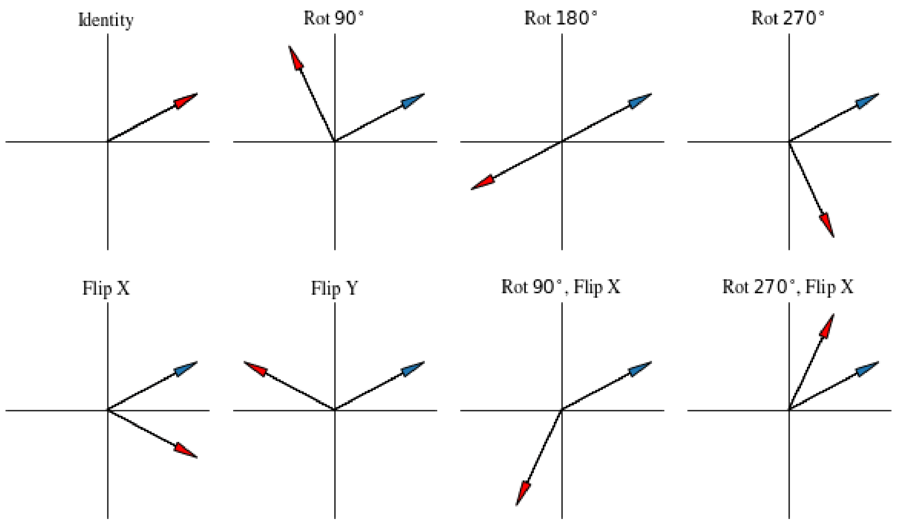


Figure 3: The elements of B_2 on the vector $\begin{bmatrix}2\\1\end{bmatrix}$. The original vector is blue and the transformed vector is red. Rotations are in degrees counterclockwise, flips are over the axis specified. The group B_d has $d!2^d$ elements, so $d = 3$ has 48 and $d = 4$ has 384 elements.

It might be a bit surprising that the group element g^{-1} appears in the definition of the action of the group on images. One way to think about it is that the pixels in the transformed image are "looked up" or "read out" from the pixels in the original untransformed image. The pixel locations in the original image are found by going back, or inverting the transformation.

Definition 20 (The group $G_{N,d}$, and its action on $k_{(p)}$-tensor images)**.** $G_{N,d}$ is the group generated by the elements of B_d and the discrete translations on the N^d-pixel lattice on the d-torus.

Remark. We view the d-torus as the quotient of the d-hypercube obtained by identifying opposite faces. The torus obtains the structure of a flat (i.e., zero curvature) Riemannian manifold this way. Because the symmetries B_d of the hypercube preserve pairs of opposite faces, they act in a well-defined way on this quotient, so we can also view B_d as a group of isometries of the torus. We choose the common fixed point of the elements of B_d as the origin for the sake of identifying the N^d pixel lattice with the group $T_{N,d} \cong (\mathbb{Z}/N\mathbb{Z})^d$ of discrete translations of this lattice; then the action of B_d on the torus induces an action of B_d on $T_{N,d}$ by automorphisms. The group $G_{N,d}$ is the semidirect product $T_{N,d} \rtimes B_d$ with respect to this action. Thus there is a canonical group homomorphism $G_{N,d} \to B_d$ with kernel $T_{N,d}$. In concrete terms, every element of $G_{N,d}$ can be written in the form $\tau \circ b$, where $b \in B_d$ and $\tau \in T_{N,d}$. Then the canonical map $G_{N,d} \to B_d$ sends $\tau \circ b$ to b.

Now that we have defined the group that we are working with, we can specify how to build convolution functions that are equivariant to $G_{N,d}$. The following theorem generalizes the Cohen and Welling paper [7] for geometric convolutions.

Theorem 1. *A $k'_{(p')}$-tensor convolution filter C produces convolutions that are equivariant with respect to the big group $G_{N,d}$ if C is invariant under the small group B_d.*

To prove this, we will first state and prove a key lemma.

Lemma 1. Given $g \in B_d$, $A \in \mathcal{A}_{N,d,k,p}$, and $C \in \mathcal{A}_{M,d,k',p'}$, the action g distributes over the convolution of A with C:

$$g \cdot (A * C) = (g \cdot A) * (g \cdot C) \tag{20}$$

Proof. Let $A \in \mathcal{A}_{N,d,k,p}$ be a geometric image, let $C \in \mathcal{A}_{M,d,k',p'}$, let $g \in B_d$, and let $\bar{\imath}$ be any pixel index of A. By Definition 19 we have

$$\begin{aligned}(g \cdot (A * C))(\bar{\imath}) &= g \cdot \big((A * C)\big(g^{-1} \cdot \bar{\imath}\big)\big) \\ &= g \cdot \left(\sum_{\bar{a} \in [-m,m]^d} A\big(g^{-1} \cdot \bar{\imath} - \bar{a}\big) \otimes C(\bar{a} + \bar{m})\right) \\ &= \sum_{\bar{a} \in [-m,m]^d} g \cdot \big(A\big(g^{-1} \cdot \bar{\imath} - \bar{a}\big) \otimes C(\bar{a} + \bar{m})\big) \\ &= \sum_{\bar{a} \in [-m,m]^d} g \cdot A\big(g^{-1} \cdot \bar{\imath} - \bar{a}\big) \otimes g \cdot C(\bar{a} + \bar{m})\end{aligned}$$

Now let $\bar{a}' = g \cdot \bar{a}$. Thus $g^{-1} \cdot \bar{a}' = g^{-1} \cdot g \cdot \bar{a} = \bar{a}$. Then:

$$\begin{aligned}(g \cdot (A * C))(\bar{\imath}) &= \sum_{\bar{a} \in [-m,m]^d} g \cdot A\big(g^{-1} \cdot \bar{\imath} - \bar{a}\big) \otimes g \cdot C(\bar{a} + \bar{m}) \\ &= \sum_{g^{-1} \cdot \bar{a}' \in [-m,m]^d} g \cdot A\big(g^{-1} \cdot \bar{\imath} - g^{-1} \cdot \bar{a}'\big) \otimes g \cdot C\big(g^{-1} \cdot \bar{a}' + \bar{m}\big) \\ &= \sum_{g^{-1} \cdot \bar{a}' \in [-m,m]^d} g \cdot A\big(g^{-1} \cdot \bar{\imath} - g^{-1} \cdot \bar{a}'\big) \otimes g \cdot C\big(g^{-1} \cdot \bar{a}' + g^{-1} \cdot \bar{m}\big) \\ &= \sum_{g^{-1} \cdot \bar{a}' \in [-m,m]^d} g \cdot A\big(g^{-1} \cdot (\bar{\imath} - \bar{a}')\big) \otimes g \cdot C\big(g^{-1} \cdot (\bar{a}' + \bar{m})\big) \\ &= \sum_{g^{-1} \cdot \bar{a}' \in [-m,m]^d} (g \cdot A)(\bar{\imath} - \bar{a}') \otimes (g \cdot C)(\bar{a}' + \bar{m}) \\ &= \sum_{\bar{a}' \in [-m,m]^d} (g \cdot A)(\bar{\imath} - \bar{a}') \otimes (g \cdot C)(\bar{a}' + \bar{m}) \\ &= ((g \cdot A) * (g \cdot C))(\bar{\imath})\end{aligned}$$

For the penultimate step, we note that $g^{-1} \cdot \bar{a}' \in [-m,m]^d$ compared to $\bar{a}' \in [-m,m]^d$ is just a reordering of those indices in the sum. Thus we have our result for pixel $\bar{\imath}$, so it holds for all pixels. □

With this lemma, the proof of Theorem 1 follows quickly.

Proof of Theorem 1. Let $A \in \mathcal{A}_{N,d,k,p}$ be a geometric image and let $C \in \mathcal{A}_{M,d,k',p'}$ be a convolution filter invariant to B_d. It is well known that convolution is equivariant to translations, and we prove it again the appendix for our definition of convolution (32). Now suppose $g \in B_d$. By Lemma 1 and the B_d-invariance of C we have:

$$g \cdot (A * C) = (g \cdot A) * (g \cdot C) = (g \cdot A) * C$$

Thus the convolution is equivariant to the generators of $G_{N,d}$, so it is equivariant to the group. □

Theorem 1 provides the foundation for building our equivariant GeometricImage-Net. Finding the set of B_d-invariant $k'_{(p')}$-tensor filters is straightforward using group averaging – see Section 5 for implementation details.

See Figure 4 and Figure 5 for the invariant convolutional filters in $d = 2$ dimensions for filters of sidelength $M = 3$ and $M = 5$ respectively. We now show some important relationships between the invariant filters of different tensor orders and parities.

Proposition 2. Let $C \in \mathcal{A}_{M,d,k',p'}$ be a B_d-invariant convolutional filter and let $\Delta \in \mathcal{A}_{M,d,2,+}$ be the geometric image with the Kronecker delta, δ, in every pixel. Then $C \otimes \Delta \in \mathcal{A}_{M,d,k'+2,p'}$ is a B_d-invariant convolutional filter.

Proof. This proof follows quickly from the B_d-invariance of the Kronecker delta, which holds because $B_d \subset O(d)$ (see Proposition 6 in the Appendix). With C and Δ defined as above and pixel $\bar{\imath}$, we have:

$$\begin{aligned}
(g \cdot (C \otimes \Delta))(\bar{\imath}) &= (g \cdot C \otimes g \cdot \Delta)(\bar{\imath}) \\
&= (g \cdot C)(\bar{\imath}) \otimes (g \cdot \Delta)(\bar{\imath}) \\
&= C(\bar{\imath}) \otimes g \cdot \Delta(g^{-1} \cdot \bar{\imath}) \\
&= C(\bar{\imath}) \otimes g \cdot \delta \\
&= C(\bar{\imath}) \otimes \delta \\
&= C(\bar{\imath}) \otimes \Delta(\bar{\imath}) \\
&= (C \otimes \Delta)(\bar{\imath})
\end{aligned}$$

□

Proposition 3. Let $C \in \mathcal{A}_{M,d,k',p'}$, $k' \geq d-1$ be a B_d-invariant convolutional filter and $\mu_1, \ldots, \mu_{d-1} \in [k']$ distinct. Then $T_{LC}(C, \mu_1, \ldots, \mu_{d-1}) \in \mathcal{A}_{M,d,k'-d+2,-p'}$ is a B_d-invariant filter of opposite parity of C.

Proof. Let C and $\mu_1, \ldots, \mu_{d-1}$ be defined as above and let $g \in B_d$. We can immediately see that $T_{LC}(C, \mu_1, \ldots, \mu_{d-1})$ is B_d-invariant by the equivariance of the Levi-Civita contraction (43), so

$$g \cdot T_{LC}(C, \mu_1, \ldots, \mu_{d-1}) = T_{LC}(g \cdot C, \mu_1, \ldots, \mu_{d-1}) = T_{LC}(C, \mu_1, \ldots, \mu_{d-1}) \ .$$

Thus we just have to verify that $T_{LC}(C, \mu_1, \ldots, \mu_{d-1}) \in \mathcal{A}_{M,d,k'-d+2,-p'}$. Since the outer product adds tensor orders and multiplies parities, at each pixel $\bar{\imath}, C(\bar{\imath}) \otimes \epsilon \in \mathcal{T}_{d,k'+d,-p'}$. Performing $d-1$ contractions reduces the tensor order by $2(d-1)$, so the resulting tensor order is $k' + d - 2(d-1) = k' + d - 2d + 2 = k' - d + 2$ as desired. □

The consequence of Propositions 2 and 3 is a natural pairing between B_d-invariant convolutional filters. See the caption of Figure 4 for further details. In practice, this allows us to dramatically reduce the number of filters we need to use in certain applications, as we will explore in Section 5.2.

4 Counting equivariant maps

With an eye to understanding the expressive power of convolution-based functions, we show how to compute the dimension of the vector space of equivariant polynomial maps of given degree.

Suppose a finite group G acts on a pair of real vector spaces V and W. Let $\mathcal{F}$ be the collection of all equivariant polynomial maps $V \to W$, and let $\mathcal{F}_\ell \subseteq \mathcal{F}$ be the homogeneous equivariant polynomials of degree ℓ. The set $\mathcal{F}_\ell$ forms a finite-dimensional real vector space whose dimension is dependent on ℓ. Thus $\dim(\mathcal{F}_\ell)$

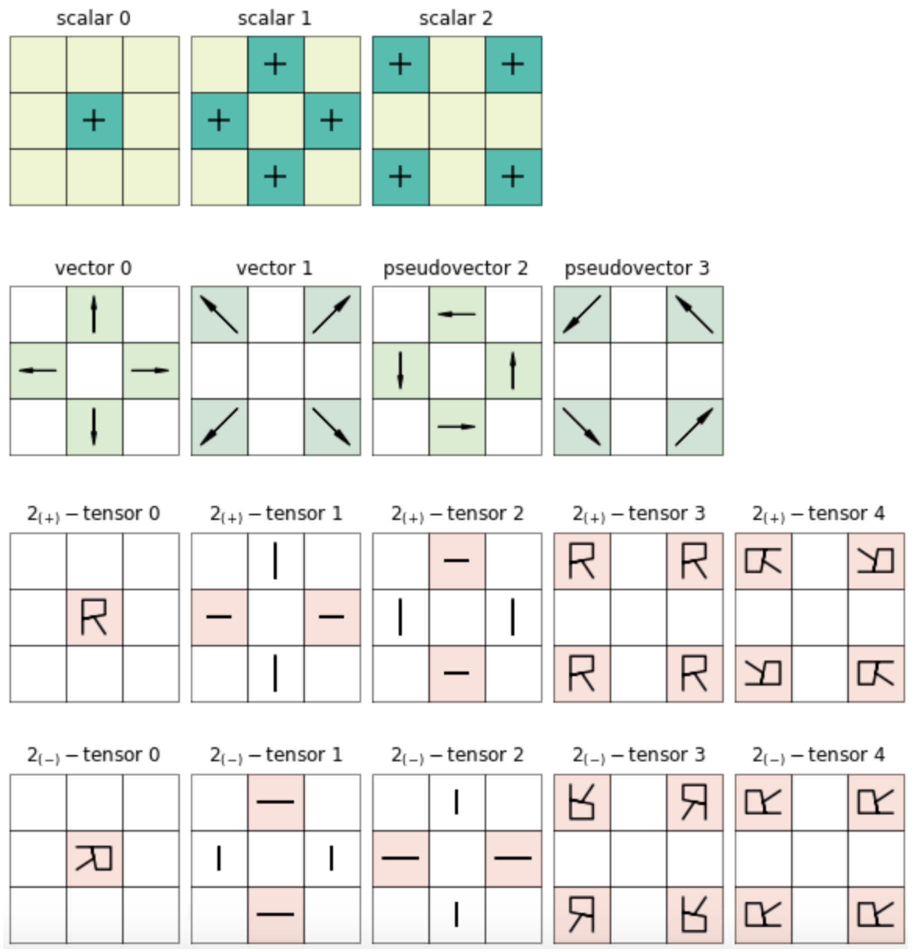


Figure 4: All the filters for $d = 2$, $M = 3$, $k' \in [0, 1, 2]$. Notes: Scalars and pseudo-scalars are shown with signed colors; where there is no symbol in the box the value is zero. The $2_{(p')}$-tensor filters are shown via the action of the tensor on an image of a letter "R"; the transformation properties of the $2_{(p')}$-tensor filters are such that these filters may not look obviously invariant to rotations. There are no invariant pseudoscalar ($0_{(-)}$-tensor) filters available at $d = 2, M = 3$. Note that scalar 0 and scalar 2 are paired with $2_{(+)}$-tensor 0 and $2_{(+)}$-tensor 3 respectively by multiplication with the Kronecker delta symbol, per Proposition 2. Likewise, if we added $2_{(+)}$-tensor 1 and $2_{(+)}$-tensor 2 together, they would be paired with scalar 1. Also note that each $1_{(+)}$-tensor filter is paired with a $1_{(-)}$-tensor filter and likewise for each $2_{(+)}$-tensor filter and $2_{(-)}$-tensor filter by Proposition 3. We don't show the $k'_{(p')}$-tensor filters at $k' > 2$ because visualizing them is difficult, even the $k' = 2$ case is potentially misleading. Note that the vector ($1_{(+)}$-tensor) filters look like pure divergence and the pseudovector ($1_{(-)}$-tensor) filters look like pure curl.

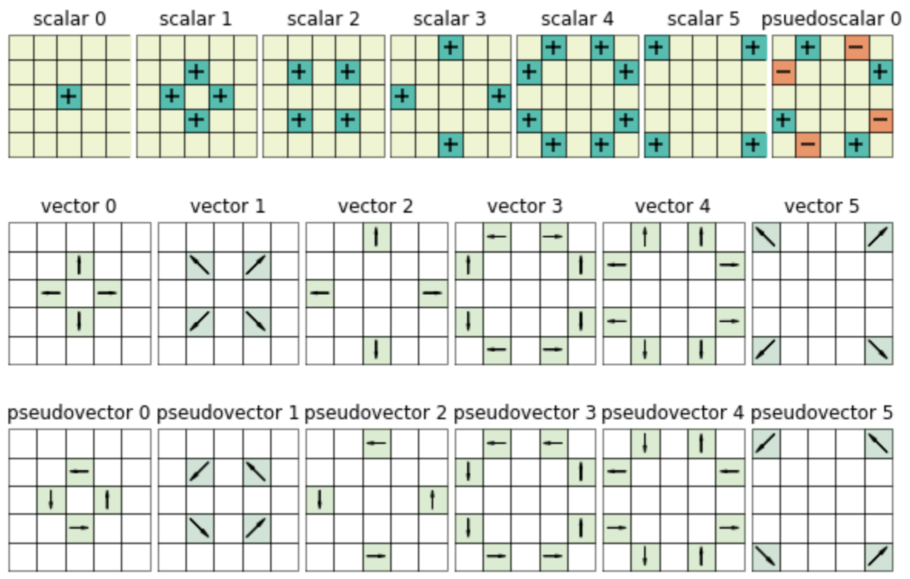


Figure 5: All the filters for $d = 2$, $M = 5$, $k' \in [0, 1]$. The symbols and colors are as in Figure 4. We don't show the $2_{(p')}$-tensor filters at $m = 2$ because there are 26 of them. At $(d, m) = (2, 2)$ a pseudoscalar filter appears. Again, the vector and pseudovectors look like pure divergence and pure curl, respectively.

forms a nonnegative integer sequence indexed by $\ell = 0, 1, 2, \ldots$. The generating function of this sequence,

$$H(\mathcal{F}, t) := \sum_{\ell \geq 0} \dim{(\mathcal{F}_\ell)} t^\ell \ , \tag{21}$$

is known as the *Hilbert series* of our set of functions. A variant [11, Remark 3.4.3] on a classical result known as *Molien's theorem* expresses this generating function as a finite sum of explicit rational functions:

$$H(\mathcal{F}, t) = \frac{1}{|G|} \sum_{g \in G} \frac{\operatorname{tr}\left(M_W\left(g^{-1}\right)\right)}{\det(I - M_V(g)t)} \ , \tag{22}$$

where $M_V(g), M_W(g^{-1})$ are matrices describing the actions of g, g^{-1} on V, W respectively. The trace in the numerator is also known as the *character* of W evaluated at g^{-1}.

Remark. The set $\mathcal{F}$ is also known as the *module of covariants* and written $(\mathbb{R}[V] \otimes W)^G$, or $\mathrm{Mor}_G(V, W)$, or $\mathrm{Mor}(V, W)^G$. In this context, the word *module* refers to the fact that the set of equivariant polynomial maps is closed under multiplication by arbitrary G-invariant polynomial functions on V as well as closed under addition. *Covariant* is another word for equivariant map, coming from classical invariant theory.

The right side of (22) is reasonable to compute in practice. To illustrate, we compute it for the group $G_{N,2}$ of Definition 20, with $V = W = \mathcal{A}_{N,2,1,+}$, the space of 2-dimensional geometric images whose pixels consist of vectors. We assume N is odd.

We first compute the character $\operatorname{tr}(M(g))$ for $g \in G_{N,2}$. This can be done explicitly by writing down a basis for $\mathcal{A}_{N,2,1,+}$ and expressing the action of each element of $G_{N,2}$ in terms of that basis. The computation is expedited by the choice of a basis in which the action of $G_{N,2}$ is *monomial*, that is, for basis vector e_i and any $g \in G_{N,d}$, we have $g \cdot e_i = \alpha e_j$, where $\alpha \in \mathbb{R}$ and e_j is some basis vector which may be the same as e_i. When this condition holds, only the basis *eigenvectors* contribute to the trace. The group B_2 acts monomially on the standard basis vectors for $\mathcal{T}_{2,1,+} \cong \mathbb{R}^2$, and it follows that $G_{N,2}$ acts monomially on a basis for $A_{N,2,1,+}$ consisting of maps $[N]^d \to \mathcal{T}_{2,1,+}$ mapping one pixel to one standard basis vector and all other pixels to zero. This situation generalizes in a straightforward fashion to higher dimensions d and higher order tensors.

Let e^0, e^1 be the standard basis for $\mathbb{R}^2$, and then for pixel index $\bar{\imath}$ and $q \in \{0,1\}$, let $\mathbf{e}^q_{\bar{\imath}} \in \mathcal{A}_{N,2,1,+}$ be the geometric image where $\mathbf{e}^q_{\bar{\imath}}(\bar{\imath}) = e^q$ and $\mathbf{e}^q_{\bar{\imath}}(\bar{\jmath}) = \vec{0}$ for all other pixel indices $\bar{\jmath} \neq \bar{\imath}$. As stated above, G acts monomially on the basis of $\mathcal{A}_{N,2,1,+}$ consisting of these images $\mathbf{e}^q_{\bar{\imath}}$. If $g \in G_{N,2}$, then $\mathbf{e}^q_{\bar{\imath}}$ is not an eigenvector for g unless g fixes the pixel $\bar{\imath}$, and even then, there is no contribution to the trace from pixel $\bar{\imath}$ unless g acts with nonzero trace on the $\operatorname{span}\left(\mathbf{e}^0_{\bar{\imath}}, \mathbf{e}^1_{\bar{\imath}}\right)$. In turn, if g does fix pixel $\bar{\imath}$, then its trace on $\operatorname{span}\left(\mathbf{e}^0_{\bar{\imath}}, \mathbf{e}^1_{\bar{\imath}}\right)$ is equal to the trace of the corresponding element $\overline{g}$ of B_2 under the canonical map $G_{N,2} \to B_2$ on $\mathbb{R}^2$. This is zero unless $\overline{g} = \pm I$ since in all other cases, $\overline{g}$ is either a $\pi/2$-rotation or a reflection. It follows that the only elements of $G_{N,2}$ with nonzero trace on $\mathcal{A}_{N,2,1,+}$ are the identity (with trace $2N^2 = \dim \mathcal{A}_{N,2,1,+}$) and the π-rotations centered at each of the N^2 pixels (each with trace -2, coming from the fixed pixel $\bar{\imath}$, where $\mathbf{e}^0_{\bar{\imath}}$ and $\mathbf{e}^1_{\bar{\imath}}$ are both negated).

Thus the only nonzero terms in the sum (22) are those with g^{-1} as just described. Conveniently, $g = g^{-1}$ in all those cases. We need to compute $\det(I - M(g)t)$ for such g. For $g = I$ we have

$$\det(I - M(g)t) = (1-t)^{2N^2} \ . \tag{23}$$

If g is a π-rotation about the pixel $\bar{\imath}$, then $\mathbf{e}^0_{\bar{\imath}}, \mathbf{e}^1_{\bar{\imath}}$ have their signs reversed, while all other pixels are transposed in pairs, say $\bar{\jmath} \leftrightarrow \bar{a}$, with the corresponding $\mathbf{e}^q_{\bar{\jmath}}$ sent to $-\mathbf{e}^q_{\bar{a}}$ and vice versa. Then the matrix $I - M(g)t$ can be written block-diagonally with two 1×1 blocks of the form $(1+t)$ for the pixel $\bar{\imath}$ that we are rotating about and $N^2 - 1$ blocks of the form

$$\begin{pmatrix} 1 & -t \\ -t & 1 \end{pmatrix} \ . \tag{24}$$

for the pixels that are being swapped. So we have

$$\det(I - M(g)t) = (1+t)^2(1-t^2)^{N^2-1}. \tag{25}$$

Putting all of this together, (22) becomes

$$H(\mathcal{F}, t) = \frac{1}{8N^2}\left(\frac{2N^2}{(1-t)^{2N^2}} + \frac{N^2(-2)}{(1+t)^2(1-t^2)^{N^2-1}}\right) \tag{26}$$

$$= \frac{1}{4}\left(\frac{1}{(1-t)^{2N^2}} - \frac{1}{(1+t)^2(1-t^2)^{N^2-1}}\right) \ . \tag{27}$$

Expanding (27) in a power series and extracting the coefficient of t^ℓ, we find that the dimension of the space of $G_{N,2}$-equivariant maps $\mathcal{A}_{N,2,1,+} \to \mathcal{A}_{N,2,1,+}$ is

$$\frac{1}{4}\left(\binom{2N^2+\ell-1}{\ell} + (-1)^{\ell+1}\sum_{j=0}^{\lfloor \ell/2 \rfloor}(\ell - 2j + 1)\binom{N^2+j-2}{j}\right) \ . \tag{28}$$

This expression evaluated for $\ell = 1, 2, 3$ is shown in Table 1.

degree ℓ	Sidelength N			
	N	3	5	7
1	$(N^2+1)/2$	5	13	25
2	$(N^4-1)/2$	40	312	1,200
3	$(2N^6+3N^4+4N^2+3)/6$	290	5,538	40,450

Table 1: The number of equivariant maps $\mathcal{A}_{N,2,1,+} \to \mathcal{A}_{N,2,1,+}$ for different values of sidelength N and degree ℓ. For degrees 1 and 2, the green cells, we were able to confirm we found all the functions. For degree 3, the pink cells, we had insufficient computer memory to confirm. For $N=3, \ell=3$ in particular, we were able to find 289 of the 290 functions by searching a subset of the candidate functions before memory limitations forced us to stop.

With the ability to explicitly count the number of $G_{N,d}$-equivariant homogeneous polynomials on geometric images, we want to know whether the operations defined in Section 2 are sufficient to characterize all these functions. Let $g_i : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k_i,p_i}$ for $i = 1, \ldots, \ell$ be a linear function on geometric images defined by the linear operations in sections 2.1 and 2.2, excluding pooling and unpooling. Let $h : \mathcal{A}_{N,d,\overline{k},\overline{p}} \to \mathcal{A}_{N,d,k'',p''}$ be a linear function defined by the same operations as the g_i functions, where $\overline{k} = \sum_{i=1}^{\ell} k_i$ and $\overline{p} = \prod_{i=1}^{\ell} p_i$. Let function $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$ be defined for all $A \in \mathcal{A}_{N,d,k,p}$:

$$f(A) = h(g_1(A) \otimes \ldots \otimes g_\ell(A)) \tag{29}$$

When $\ell = 1$, we will only do $f(A) = h(A)$ rather than $f(A) = h(g_1(A))$.

We conjecture that these steps will allow us to construct all equivariant maps of any degree. To test this conjecture, we performed the following experiments to count the number of linear, quadratic, and cubic homogeneous polynomials from vector images to vector images. First we constructed all the B_d-invariant $k'_{(p')}$-tensor filters for $k' = 1, 2$ and $p' = +1$ and used those filters to construct all the homogeneous polynomials according to (29). We then generated a random vector image and applied all the functions to that image, and we want to know whether those output images are linearly independent. Thus we flattened all the resulting images into a giant matrix and performed a singular value decomposition; the number of non-zero singular values gives us the number of linearly independent functions. In the higher order polynomial cases we have to apply the various functions on multiple images to ensure separation. The results are given in Table 1.

5 GeometricImage-Net Architectures

Our GeometricImage-Net model seeks to learn some function $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$. The problem determines d, and therefore the groups B_d and $G_{N,d}$. After fixing these initial parameters, the modeler must decide the size, number, and type of layers that are described below. The first choice is the attributes of the convolution filters: size M, tensor order k', and parity p', all of which may be a single value or multiple values.

A complete set of B_d-invariant $k'_{(p')}$-tensor filters can be found by group averaging. We first construct all group operators for the B_d group by iterating the generators until the group is closed under all products of operators. The set of possible geometric filters is a vector space of dimension $M^d d^{k'}$, so we can pick a basis of that many elements where each basis element C_i has exactly one component of the tensor in a single pixel set to 1, and all other values are 0. Each of these basis elements is then group-averaged by applying all group operators and averaging the results:

$$\widetilde{C}_i = \frac{1}{|B_d|} \sum_{g \in B_d} g \cdot C_i \ , \tag{30}$$

where $|B_d|$ is the number of group elements. The results of those group averages are unpacked into a matrix and the singular value decomposition is then run to find an "eigen-set" of orthogonal, non-zero filters. After the SVD, the filters can be normalized however seems appropriate. We normalized the filters such that the magnitudes of the non-zero filter values are as close to unity as possible, and the $k = 1$ filters are also reoriented such that non-zero divergences were set to be positive, and non-zero curls were set to be counter-clockwise. See Figure 4 and Figure 5 for the invariant convolutional filters in $d = 2$ dimensions for filters of sidelength $M = 3$ and $M = 5$ respectively. With the set of invariant filters in hand, we may build our equivariant neural networks using convolution layers, contraction layers, outer product layers, and nonlinear activation layers.

5.1 Architecture Components

We think of each building block of our architecture as a layer whose input and output is a set of images grouped by tensor order and parity. The reason for grouping is two-fold: we can only add geometric images that share tensor order and parity, and we can more efficiently batch our operations in JAX when the shapes match. The operation of each layer is either convolution, contraction, taking outer products, or applying a nonlinear activation function.

A convolution layer takes a set of images and a set of convolution filters. For each filter, we take a weighted sum of the images of a particular tensor order and parity and apply the convolution[1] on that sum with that filter. Unlike a traditional CNN where the filters are parameterized, our filters are fixed to enforce the equivariance, and the weights of the weighted sums are the parameters. A convolution layer can also have an optional dilation where the filters are dilated before convolving. Dilations are helpful for expanding the effective size of the filters without having to calculate the invariant filters for larger M, which grows quickly; see [12] for a description of dilated convolution. If we use filters with tensor order greater than 0, the tensor order of the images will continue to grow as we apply convolutions. Thus we need a way to reduce the tensor order – we do this with the contraction layer.

Given an input layer and a desired tensor order, the contraction layer performs all unique contractions (see Contraction Properties (35)(36)) to get the layer to that tensor order. We always end the neural network with a contraction layer to return the images to the proper tensor order. Since contractions can only reduce the tensor order by multiples of 2, the last convolution layer before the final contraction must result in images of order $k'' + 2n$ for any $n \in \mathbb{N}$. We also may include contraction layers after each convolution to cap the tensor order of each layer to avoid running out of memory as the tensor order grows. In practice, $k = 5$ seems to be a good max tensor order.

An outer product layer takes a set of images and a degree and computes the full polynomial of all the images with each other using the outer product of geometric images, up to the specified degree. Typically, this will result in a combinatorial blowup of images; we can take parameterized sums along the way to reduce the

[1]The geometric convolution package is implemented in JAX, which in turn uses TensorFlow XLA under the hood. This means that convolution is actually cross-correlation, in line with how the term in used in machine learning papers. For our purposes this results in at most a sign change, which will come out in the parameterization.

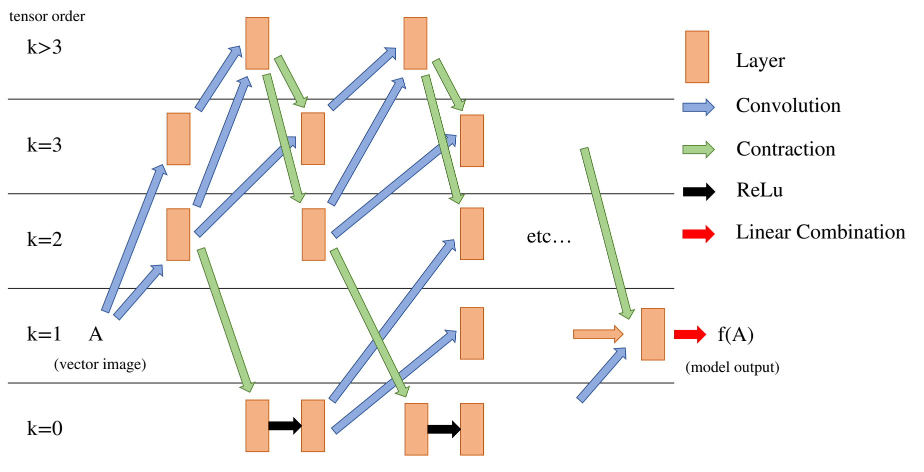


Figure 6: One possible architecture for a GI-Net that maps a vector image to a vector image when using convolution filters with tensor order $k' \in \{1, 2\}$, parity $p' = +1$, max order $k = 3$, and ReLu nonlinearities. Each layer is a block of multiple images that share tensor order. The blue arrows represent convolutions and raise the tensor order by 1 or 2. Contractions are applied when the tensor order goes above 3 to bring it down to 2 or 3, and when contracting to $k = 0$ in order to apply the ReLu. This process continues until the final step where the only layer is order $k = 1$, which is then combined using a parameterized linear combination.

number of images created. We could also do a smaller set of products if we have some special domain knowledge. However, in practice it is usually better to use nonlinear activation functions, as is standard in machine learning.

The final type of layer is a nonlinear activation layer. In order to maintain equivariance, we can either apply a nonlinearity to a scalar, or scale our tensors by a nonlinearity applied to the norm of the tensor [42]. For this paper, we used the first strategy. We apply all possible contractions to all even tensor order images to reduce them to scalars, then apply the nonlinearity. Any typical nonlinearity works – ReLU, leaky ReLu, sigmoid, etc. This layer will result in scalar images, which will then grow in order again as we apply more convolution layers.

5.2 Architecture Efficiency

Without specialized knowledge of what B_d-invariant convolution filters are relevant for our problem, we want to use all the filters at a specified tensor order in our convolution layers. Thus we can improve the efficiency of the GI-Net by determining what filters are redundant. The first result follows from Proposition 2 and says that we may omit the $k'_{(p')}$-tensor filters if we are using the $(k'+2)_{(p')}$-tensor filters and we are taking all contractions later.

Proposition 4. Let $\mathcal{F}$ be the set of functions $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$ where each f is a convolution with a B_d-invariant $k'_{(p')}$-tensor filter. Let $\mathcal{G}$ be the set of functions $g : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$ where each g is a convolution with a B_d-invariant $(k'+2)_{(p')}$-tensor filter followed by a contraction. Then $\mathcal{F} \subseteq \mathcal{G}$.

See Appendix A for the proof. This proposition can be repeatedly applied so that if we conclude a GI-Net by taking all unique contractions, then we need only include filters of tensor order k' and $k'-1$ to include all smaller tensor orders as well. The next result says that if the input and output parities of our network are equal, we may omit the $k'_{(-)}$-tensor filters if we are using the $(k'+d)_{(+)}$-tensor filters and taking all contractions later.

Proposition 5. Let $\mathcal{F}$ be the set of functions that preserve parity $f:\mathcal{A}_{N,d,k,p}\to\mathcal{A}_{N,d,k'',p}$ where each f is a convolution with a negative-parity $k'_{(-)}$-tensor filter followed by a Levi-Civita contraction. Let $\mathcal{G}$ be the set of functions that preserve parity $g:\mathcal{A}_{N,d,k,p}\to\mathcal{A}_{N,d,k'',p}$ where each g is a convolution with a positive-parity $(k'+d)_{(+)}$-tensor followed by $d-1$ contractions. Then $\mathcal{F}\subseteq\mathcal{G}$.

See Appendix A for the proof. We will employ these results in our numerical experiments to dramatically reduce the number of filters required.

6 Numerical Experiments

Code to reproduce these experiments and build your own GI-Net is available at `https://github.com/WilsonGregory/GeometricConvolutions`. The code is built in Python using JAX.

The most natural problems for this model are those that we expect to obey the symmetries of the group $G_{N,d}$. We present two problems from physics that despite their simplicity, exhibit the powerful generalization properties of the equivariant model even in cases where we have minimal training points.

First, suppose we have as input a scalar image of point masses, and we want to learn the gravitational vector field induced by these masses. For this problem, we will work in two dimensions with image sidelength of 16, so the GI-Net is learning a function $f:\mathcal{A}_{16,2,0,+}\to\mathcal{A}_{16,2,1,+}$. To generate the data, we sampled the pixel locations uniformly without replacement 5 times to be point masses, and then we set their masses to be a uniform value between 0 and 1.

For a second problem, we consider several point charges in a viscous fluid. All the point charges have charge +1, so they repel each other. The position of the charges in the fluid would be described by an ordinary differential equation, and we can approximate that using Euler's method:

$$x_i(t+\Delta t)=x_i(t)+\Delta t V(x_i,t)\ , \tag{31}$$

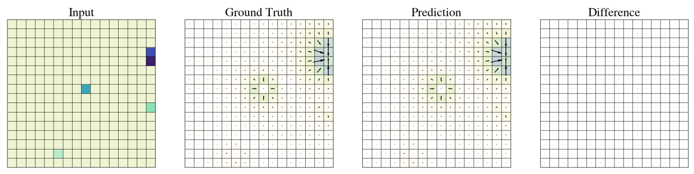
where x_i is a point, Δt is one time step, and $V(x_i,t)$ is the vector field at time t induced by all particles other than x_i. We iterate this system some number of steps T, and the learning problem is the following: Given the initial charge field, can we predict the charge field after step T? For this toy problem we will again use an image in two dimensions of sidelength 16, so the function we are trying to learn is $f:\mathcal{A}_{16,2,1,+}\to\mathcal{A}_{16,2,1,+}$.

We took several precautions to make the data well behaved. Since the particles move freely in $\mathbb{R}^2$ but we learn on a discrete vector field grid, the vectors act erratic when the charge passes very closely to the center of the pixel. Thus we applied a sigmoid transform on the charge field vectors on the input and output:

$$Q(\vec{v},s)=\left(\frac{1}{1+e^{-\frac{\|\vec{v}\|}{s}}}-\frac{1}{2}\right)\left(\frac{\vec{v}}{\|\vec{v}\|}\right)\ ,$$

where $\|\cdot\|$ is the usual Euclidean norm and s is a parameter that we set to 0.2. One advantage of learning on the charge field rather than the particles themselves is that vector field is discrete, but the vectors reflect the exact particle locations. However, if two particles start very close, it will appear that there is only a single particle on the charge vector field. To alleviate this problem, we iterated one step of

(a)



(b)

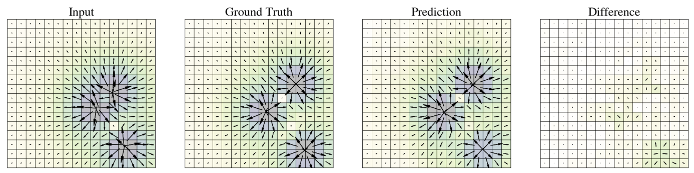


Figure 7: One test point each for the (a) gravitational field problem and the (b) moving point charge problem. The first two columns show the input and output to the problems respectively. The third column shows the model predicted output, and the fourth column shows the difference between the ground truth and the predicted output. To put the numerical results in context, the loss for the top row is 0.0177, while the loss for the bottom row is 2.061.

Euler's method, and treated that as the input to the neural network. Additionally, we initialized points within the central 8×8 grid rather than the full 16×16 grid so that the charges would be unlikely to leave the bounds of the grid by step T. See Figure 7 for example inputs and outputs for the two problems.

6.1 Architectures

For the gravity problem, the architecture has 3 convolution layers followed by all contractions that reduce to $k = 1$, and then a parameterized linear combination of the resulting images. The second convolution layer uses dilations that range from 1 to 15, and the third convolution layer uses dilations from 1 to 7. For our loss we use the root mean squared error (RMSE). The baseline architecture is built to have similar structure with 3 convolution layers with the same dilations, but also have a number parameters on the same order of magnitude. We treat the dilations as creating a separate channel. The sequence of channel depth is the following:

$$1 \to 2 \to (15 * 2) \to (7 * 2) \to 2$$

To get from the output of the 3rd convolution which is 2 filters across 7 dilations, we take a parameterized sum across the dilations to get an image with 2 channels, which is the number of channels we need for a vector image. See Table 2 for additional info about the filters and number of parameters.

The moving charges problem is more difficult, so the architecture complexity reflects that. We have 9 convolution layers, with dilations of $1, 2, 4, 2, 1, 1, 2, 1, 1$ in that order. Each convolution is followed by a nonlinear activation layer with a Leaky ReLu with negative slope of 0.01. This non-linearity seemed to perform best among the ones we tried. We then finish with the usual contraction layer and linear combination, and our loss is again the RMSE. For the baseline model, we use identical number of convolution layers, dilations, nonlinearities, and loss. The only difference is that we only use scalar filters, so we increase the depth of each layer

problem	model	M	k'	p'	# layers	depths	# params
gravity	GI-Net	3	0,1	+1	3	1	3,085
	baseline	3	0		3	$2{,}15*2, 7*2$	4,345
charge	GI-Net	3	1,2	+1	9	1	22,986
	baseline	3	0		9	20,...,20,2	25,920

Table 2: Comparison of the different model architectures used for the two problems. The values of M, k', p' represent the sidelength, tensor order, and parity of the convolutional filters. The baseline models do not have a parity because those filters are learned rather than fixed ahead of time.

to 20 except for the final output layer which must have a depth of 2 because we are learning a vector image.

6.2 Training

For all models, we trained the network using stochastic gradient descent with the Adam optimizer and an exponential learning rate decay that started at 0.005, had transition steps equal to the number of batches per epoch, and a decay of 0.995. The one exception was the baseline model for the moving charges problem where we started with a learning rate of 0.001. These values were found with a limited grid search. For both problems and both models we initialized the parameters Gaussian noise with mean 0 and standard deviation 0.1.

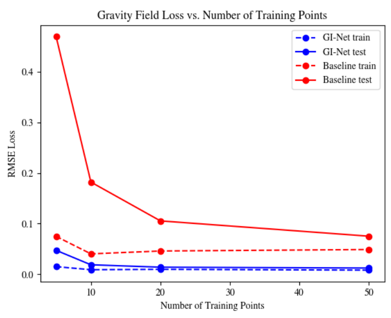
For the gravitational field problem, we created a test set of 10 images, a validation set of 5 images, and training sets ranging in size from 5 to 50 images. For the moving charges problem we created a test set of 10 images, a validation set of 10 images, and training sets ranging in size from 5 to 100 images. We used training batch sizes equal to $\frac{1}{5}^{th}$ the training set size. For all models we trained them until the error on the validation set had not improved in 20 epochs.

6.3 Results

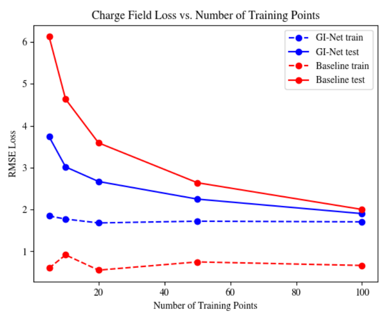
Given sufficient data, both the GI-Net and the baseline model are able to perform well on the test data set. The RMSE carries very little information without further context, but we can see from the examples in Figure 7 that low error corresponds to only small differences between the ground truth output and the predicted output. By comparing the GI-Net to our simple CNN baseline we can see the advantages of the equivariant model.

In Figure 8(a) we can see that with only 10 data points, the test error of the GI-Net is almost equal to the training error, suggesting that the model is able to generalize well from just a few small examples. By contrast, the baseline model requires at least 50 training points to get its test error as close to its training error. Additionally, even when the baseline model has enough points, its error is still higher overall compared to the GI-Net.

Likewise, in Figure 8(b) the gap between the test error and training error for the baseline model is much larger than the same gap for the GI-Net. In this case, the GI-Net and the baseline model reach the same test error when training off 100 points, despite the baseline model having smaller training error. This again suggests that the GI-Net does a better job generalizing, especially in the small training data set regime.



(a) Gravitational Field Loss



(b) Charge Field Loss

Figure 8: Comparison between baseline model and GI-Net as a function of the number of points in the training data set. The GI-Net have smaller testing loss when the training data set is small, even when the baseline model achieves better training error in the case of the moving charges problem.

7 Related work (Summary)

Restricting the learning to a hypothesis class where all the functions are equivariant with respect to some group action is a is a powerful tool widely used in machine learning. Engineering a model to be equivariant, as we have done, improves its generalization and accuracy (see for instance [16],[15],[45]) and is a powerful remedy against the lack of data (see [47]).

Equivariant networks, in certain cases, can approximate any continuous invariant function (see [51], [13], [3], [30]). Our maps can be used in a similar way to approximate continuous non-linear equivariant maps.

The interest in equivariant maps for machine learning has lead to a huge garden of methods for various kinds of data and to the reinvention of the same algorithms in different application domains. Irreducible representations or invariant theory are sometimes used to parameterize the space of equivariant functions. The first work using representation theory for invariant and equivariant neural networks was the paper of Wood et al. [48] from 1996. More recent incarnations of these ideas include the works of [35], [8], [7], [6], [10] and [17].

One can use classical invariant theory to compute the generators of the algebra of invariant polynomials. For instance, in [2] we show how to use the generators of the algebra of invariant polynomials to produce a parameterization of equivariant functions for certain groups and actions. This approach is inspired by the physical sciences, where the data is subject to rules coming from coordinate freedoms and conservation laws. In previous work [42, 43, 50] we used these geometric rules to develop modified machine-learning methods that are restricted to exactly obey group-theoretic equivariances in physics contexts. The same idea has explored in [23], [20], [50].

The most common method to design equivariant maps is via group convolution, on the group or on the homogeneous space where the features lie. Our generalization of convolution replaces this scalar product of vectors by the outer product of tensors. This approach is similar to the one of Brandstetter et al. [4], which replace it by the geometric product of multivector inputs and multivector filters of a Clifford Algebra, and Rose Yu et. al [46], which replace convolution with a single kernel for convolution with a kernel depending on the group element.

From a Fourier perspective, we generalize the results proved in [29] about that group convolutional structure is a sufficient and a necessary condition for equivariance to the action of a compact group.

See appendix B for a more in depth description of the mathematical details of the related work.

8 Discussion

One of this work's primary strengths, the fact that we are working with tensors, is also one of its key limitations. Building convolutions with tensor products and contractions gives us the flexibility to write a functions on whatever wacky images the physicists have dreamt up. One hundred years of differential geometry has given us the tools to prove results about the equivariance of our model and given us some confidence in its expressive power. Yet those same tensors have made it challenging to show numerically that we have characterized all the equivariant homogeneous polynomials because of the exponential growth in memory requirements as the tensor order grows. Indeed, it is still an open question to understand the full expressive power of the model outside the few cases we were able to test in Section 4. The memory constraints of large order tensors are also a challenge in our numerical experiments in Section 6, but fortunately it appears that capping the maximum tensor order pays dividends in memory management without much associated loss in performance.

Another point of tension in this work is that the index summation rules and geometric operations are motivated by the continuous Euclidean group, but the images that we work with are d-cube lattices of points which make it more natural to work with the discrete group $G_{N,d}$, a subgroup of the Euclidean group. Using the group $G_{N,d}$ allows us to sidestep voxelizing our images for rotations other than 90 degrees, but of course we expect approximate equivariance to any rotation to appear in nature. A picture of a banana rotated 17 degrees is still a picture of a banana, and it is only an approximate equivariance because the discrete nature of our image makes it approximate. There are other possible image representations that might create more continuous concepts of images. For example, if the data is on the surface of a sphere, it could be represented with tensor spherical harmonics, and be subject to transformations by a continuous rotation group.

There are also interesting future directions of research in various applications more complex than the simple examples we gave in Section 6. The model could be used on images such as those in Figure 1, or perhaps in more standard image analysis problems to see how it compares to classical techniques. The proof is in the pudding, as they say.

Acknowlegements: It is a pleasure to thank Roger Blandford (Stanford), Drummond Fielding (Flatiron), Leslie Greengard (Flatiron), Ningyuan (Teresa) Huang (JHU), Kate Storey-Fisher (NYU), and the Astronomical Data Group at the Flatiron Institute for valuable discussions and input. This project made use of Python 3 [40], numpy [24], matplotlib [28], and cmastro [36]. All the code used for making the data and figures in this paper is available at `https://github.com/WilsonGregory/GeometricConvolutions`.

Funding: Maria Teresa Arias was supported by H2020-MSCA-RISE-2017, Project 777822, and from Grant PID2019-105599GB-I00, Ministerio de Ciencia, Innovación y Universidades, Spain.

References

[1] *Climate Data Guide.* UCAR, 2015.

[2] Ben Blum-Smith and Soledad Villar. Machine learning and invariant theory, 2022.

[3] Georg Bökman, Fredrik Kahl, and Axel Flinth. Zz-net: A universal rotation equivariant architecture for 2d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10976–10985, 2022.

[4] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K. Gupta. Clifford neural layers for pde modeling, 2022.

[5] Gregory S. Chirikjian and Alexander B. Kyatkin. *Engineering applications of noncommutative harmonic analysis: With emphasis on rotation and motion groups.* CRC PRESS, 2021.

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

[7] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[8] Taco S. Cohen and Max Welling. Steerable cnns. 2016.

[9] Planck Collaboration. Planck 2015 results – i. overview of products and scientific results. *A&A*, 594:A1, 2016.

[10] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020.

[11] Harm Derksen and Gregor Kemper. *Computational invariant theory.* Springer, 2015.

[12] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016.

[13] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *arXiv:2010.02449*, 2020.

[14] Albert Einstein. Die Grundlage der allgemeinen Relativitätstheorie. *Annalen der Physik*, 354(7):769–822, January 1916.

[15] Bryn Elesedy. Provably strict generalisation benefit for invariance in kernel methods. *arXiv preprint arXiv:2106.02346*, 2021.

[16] Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. *arXiv preprint arXiv:2102.10333*, 2021.

[17] Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. Spin-weighted spherical cnns, 2020.

[18] G. B. Folland. *A course in abstract harmonic analysis.* CRC Press, 2016.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[20] Ben Gripaios, Ward Haddadin, and Christopher G Lester. Lorentz- and permutation-invariants of particles. *Journal of Physics A: Mathematical and Theoretical*, 54(15):155201, mar 2021.

[21] Victor Guillemin and Alan Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.

[22] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759, apr 2005.

[23] Ward Haddadin. Invariant polynomials and machine learning. *arXiv preprint arXiv:2104.12733*, 2021.

[24] Charles R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[26] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[27] James E Humphreys. *Reflection groups and Coxeter groups*. Number 29. Cambridge university press, 1990.

[28] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[29] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[30] Wataru Kumagai and Akiyoshi Sannai. Universal approximation theorem for equivariant maps by group cnns. *arXiv preprint arXiv:2012.13882*, 2020.

[31] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[32] Valery I Levitas, Mehdi Kamrani, and Biao Feng. Tensorial stress–strain fields and large elastoplasticity as well as friction in diamond anvil cell up to 400 gpa. *NPJ Computational Materials*, 5(1):1–11, 2019.

[33] S. Lossow, M. Khaplanov, J. Gumbel, Jacek Stegman, Georg Witt, Peter Dalin, Sheila Kirkwood, F. Schmidlin, K. Fricke, and U.A. Blum. Middle atmospheric water vapour and dynamics in the vicinity of the polar vortex during the hygrosonde-2 campaign. *Atmospheric Chemistry and Physics*, 9, 07 2009.

[34] Ib H Madsen and Jxrgen Tornehave. *From calculus to cohomology: de Rham cohomology and characteristic classes*. Cambridge university press, 1997.

[35] Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis. Correspondence-free structure from motion. *Int. J. Comput. Vision*, 75(3):311–327, dec 2007.

[36] Adrian M. Price-Whelan. cmastro: colormaps for astronomers. `https://github.com/adrn/cmastro`, 2021.

[37] M. M. G. Ricci and Tullio Levi-Civita. Méthodes de calcul différentiel absolu et leurs applications. *Mathematische Annalen*, 54(1):125–201, 1900.

[38] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019.

[39] Kip S. Thorne and Roger D. Blandford. *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*. Princeton University Press, 2017.

[40] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[42] Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

[43] Soledad Villar, Weichi Yao, David W Hogg, Ben Blum-Smith, and Bianca Dumitrascu. Dimensionless machine learning: Imposing exact units equivariance. *arXiv preprint arXiv:2204.00887*, 2022.

[44] Di Wang and Xi Zhang. Modeling of a 3d temperature field by integrating a physics-specific model and spatiotemporal stochastic processes. *Applied Sciences*, 9(10), 2019.

[45] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations*, 2021.

[46] Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics, 2022.

[47] Rui Wang, Robin Walters, and Rose Yu. Data augmentation vs. equivariant networks: A theory of generalization on dynamics forecasting. *arXiv preprint arXiv:2206.09450*, 2022.

[48] Jeffrey Wood and John Shawe-Taylor. Representation theory and invariant neural networks. *Discrete Applied Mathematics*, 69(1):33–60, 1996.

[49] Yinshuang Xu, Jiahui Lei, Edgar Dobriban, and Kostas Daniilidis. Unified fourier-based kernel and nonlinearity design for equivariant networks on homogeneous spaces, 2022.

[50] Weichi Yao, Kate Storey-Fisher, David W. Hogg, and Soledad Villar. A simple equivariant machine learning method for dynamics based on scalars, 2021.

[51] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *arXiv:1804.10306*, 2018.

[52] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.

A Propositions and Proofs

This section provides propositions and their proofs that are necessary for some of the results earlier in the paper. In many of these proofs we will show that the property holds for some arbitrary pixel index $\bar{\imath}$, so it must hold for all pixels. First we have two well known results from tensor analysis.

Proposition 6. The Kronecker delta, Definition 4, is invariant to the group B_d.

Proof. Let $g \in B_d$ with matrix representation $M(g)$. The Kronecker delta δ is a $2_{(+)}$-tensor so the action of g is by conjugation. Thus,

$$g \cdot \delta = M(g)\delta M(g)^T = M(g)M(g)^T = \delta$$

since the Kronecker delta is also the $d \times d$ identity matrix and the matrix representations of B_d are orthogonal. □

Proposition 7. The Levi-Civita symbol, Definition 5, is invariant to the group B_d.

Proof. The Levi-Civita tensor $\epsilon \in (\mathbb{R}^d)^{\otimes d}$ is defined so as to satisfy the identity

$$\epsilon^\sigma = \text{sgn}(\sigma)\epsilon,$$

where $\sigma \in S_d$ is a permutation of the indices (cf. Definition 7). Thus it is an alternating tensor of order d. It is well-known (e.g., [21, p. 160] or [34, p. 13]) that the subspace of these is (one-dimensional and) stable under the action of $GL(\mathbb{R}^d)$ on $(\mathbb{R}^d)^{\otimes d}$ given by linear extension of $M\cdot(u_1\otimes\cdots\otimes u_d) := (Mu_1)\otimes\cdots\otimes(Mu_d)$, where $M \in GL(\mathbb{R}^d)$ is an arbitrary invertible matrix, and that M acts on this subspace by multiplication by $\det(M)$. Thus

$$M \cdot \epsilon = \det(M)\epsilon,$$

where the action on the left is the one just described. Using instead the action under consideration throughout this paper, i.e., the action of $O(d) \subset GL(\mathbb{R}^d)$ defined by equation (1) and Definition 1, we get an additional determinant factor on the right side because ϵ is a $d_{(-)}$-tensor. In other words, $g \cdot \epsilon = \det(M(g))^2\epsilon$. Since $\det(M(g))^2 = 1$ for all $g \in O(d)$, we conclude that $g \cdot \epsilon = \epsilon$. □

Next we state several properties of geometric convolution that are well known for the general mathematical definition of convolution.

Properties (Convolution)**.** Given $A, B \in \mathcal{A}_{N,d,k,p}, C, S \in \mathcal{A}_{M,d,k',p'}, \tau \in (\mathbb{Z}/N\mathbb{Z})^d$ and $\alpha, \beta \in \mathbb{R}$, then the following properties hold:

1. The convolution operation is translation equivariant:

$$(L_\tau A) * C = L_\tau(A * C) \tag{32}$$

2. The convolution operation is linear in the geometric image:

$$(\alpha A + \beta B) * C = \alpha(A * C) + \beta(B * C) \tag{33}$$

   It is also linear in the filters:

$$A * (\alpha C + \beta S) = \alpha(A * C) + \beta(A * S) \tag{34}$$

Proof. First we will prove (32). Let A, C, and τ be as above and let $\bar{\imath}$ be a pixel index of $L_\tau A * C$. Then:

$$\begin{aligned}
(L_\tau A * C)(\bar{\imath}) &= \sum_{\bar{a}\in[-m,m]^d} (L_\tau A)(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) \\
&= \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}-\tau) \otimes C(\bar{a}+\bar{m}) \\
&= \sum_{\bar{a}\in[-m,m]^d} A((\bar{\imath}-\tau)-\bar{a}) \otimes C(\bar{a}+\bar{m}) \\
&= (A * C)(\bar{\imath}-\tau) \\
&= L_\tau(A * C)(\bar{\imath})
\end{aligned}$$

Now we will prove (33). Let A, B, C, α, and β be as above and let $\bar{\imath}$ be a pixel index of $(\alpha A + \beta B) * C$. Then:

$$\begin{aligned}
((\alpha A + \beta B) * C)(\bar{\imath}) &= \sum_{\bar{a}\in[-m,m]^d} (\alpha A + \beta B)(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) \\
&= \sum_{\bar{a}\in[-m,m]^d} (\alpha A(\bar{\imath}-\bar{a}) + \beta B(\bar{\imath}-\bar{a})) \otimes C(\bar{a}+\bar{m}) \\
&= \sum_{\bar{a}\in[-m,m]^d} \alpha A(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) + \beta B(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) \\
&= \alpha \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) + \beta \sum_{\bar{a}\in[-m,m]^d} B(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) \\
&= \alpha(A * C)(\bar{\imath}) + \beta(B * C)(\bar{\imath})
\end{aligned}$$

Now we will prove (34). Let A, C, S, α, and β be as above and let $\bar{\imath}$ be a pixel index. Then:

$$\begin{aligned}
(A * (\alpha C + \beta S))(\bar{\imath}) &= \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes (\alpha C + \beta S)(\bar{a}+\bar{m}) \\
&= \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes \alpha C(\bar{a}+\bar{m}) + A(\bar{\imath}-\bar{a}) \otimes \beta S(\bar{a}+\bar{m}) \\
&= \alpha \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes C(\bar{a}+\bar{m}) + \beta \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes S(\bar{a}+\bar{m}) \\
&= \alpha(A * C)(\bar{\imath}) + \beta(A * S)(\bar{\imath})
\end{aligned}$$

Thus we have our result. □

Now we will state several properties of the contraction.

Properties (Contraction)**.** Given $A, B \in \mathcal{A}_{N,d,k,p}, S \in \mathcal{A}_{N,d,k',p'}, C \in \mathcal{C}_{M,d,k',p'}$, $g \in G_{N,d}, \mu, \nu, \rho, \sigma \in [k]$ all distinct, and $\alpha, \beta \in \mathbb{R}$, then the following properties hold:

1. Contraction indices can be swapped:

$$T(A, \mu, \nu) = T(A, \nu, \mu) \tag{35}$$

2. Contractions on distinct indices commute:

$$T_M(A, (\mu, \nu), (\rho, \sigma)) = T_M(A, (\rho, \sigma), (\mu, \nu)) \tag{36}$$

3. Contractions are equivariant to $G_{N,d}$:

$$g \cdot T(A, \mu, \nu) = T(g \cdot A, \mu, \nu) \tag{37}$$

4. Contractions are linear functions:
$$T(\alpha A+\beta B,\mu,\nu)=\alpha\, T(A,\mu,\nu)+\beta\, T(B,\mu,\nu) \tag{38}$$
5. Contractions commute with the outer product. If $\mu,\nu\in[k],\mu\neq\nu$ then:
$$T(A\otimes S,\mu,\nu)=T(A,\mu,\nu)\otimes S \tag{39}$$
Otherwise, if $\mu,\nu\in\{k+1,\ldots,k+k'\},\mu\neq\nu$ then:
$$T(A\otimes S,\mu,\nu)=A\otimes T(S,\mu-k,\nu-k) \tag{40}$$
6. Contractions commute with convolutions. If $\mu,\nu\in[k]$ distinct then:
$$T(A*C,\mu,\nu)=T(A,\mu,\nu)*C \tag{41}$$
Otherwise, if $\mu,\nu\in\{k+1,\ldots,k+k'\}$ distinct then:
$$T(A*C,\mu,\nu)=A*T(C,\mu-k,\nu-k) \tag{42}$$

Proof. Equation (35) follows directly from the definition, 6. Now we will prove (36). Let A,μ,ν,ρ, and σ be defined as above, let $\bar\imath$ be a pixel index of A, and let $a\in\mathcal{T}_{d,k,p}$ be the tensor at that index. Since contractions are applied the same to all pixels, it suffices to show that this proposition is true for this pixel. The result then follows quickly from the definition:
$$\begin{aligned}[T_M(a,(\mu,\nu),(\rho,\sigma))]_{\{i_1,\ldots,i_k\}\setminus\{i_\mu,i_\nu,i_\rho,i_\sigma\}} &= [\delta]_{i_\rho,i_\sigma}\left([\delta]_{i_\mu,i_\nu}[a]_{i_1,\ldots,i_k}\right)\\ &= [\delta]_{i_\mu,i_\nu}[\delta]_{i_\rho,i_\sigma}[a]_{i_1,\ldots,i_k}\\ &= [\delta]_{i_\mu,i_\nu}\left([\delta]_{i_\rho,i_\sigma}[a]_{i_1,\ldots,i_k}\right)\\ &= [T_M(a,(\rho,\sigma),(\mu,\nu))]_{\{i_1,\ldots,i_k\}\setminus\{i_\mu,i_\nu,i_\rho,i_\sigma\}}\end{aligned}$$

Next we will prove (37). Let $A,\mu,$ and ν be defined as above and let $\bar\imath$ be a pixel of A. First we will show that contractions are equivariant to translations. Let $\tau\in(\mathbb{Z}/N\mathbb{Z})^d$. Then
$$\begin{aligned}T(L_\tau A,\mu,\nu)(\bar\imath) &= T((L_\tau A)(\bar\imath),\mu,\nu)\\ &= T(A(\bar\imath-\tau),\mu,\nu)\\ &= T(A,\mu,\nu)(\bar\imath-\tau)\\ &= (L_\tau T(A,\mu,\nu))(\bar\imath)\end{aligned}$$

Thus contractions are equivariant to translations. Now we will show that contractions are equivariant to B_d. Let $g\in B_d$, and denote $A(g^{-1}\bar\imath)=a$. Then by equation (4) we have:
$$\begin{aligned}T(g\cdot A,\mu,\nu)(\bar\imath) &= T((g\cdot A)(\bar\imath),\mu,\nu)\\ &= T\big(g\cdot A(g^{-1}\cdot\bar\imath),\mu,\nu\big)\\ &= T(g\cdot a,\mu,\nu)\\ &= [\delta]_{i_\mu,i_\nu}[g\cdot a]_{i_1,\ldots,i_\mu,\ldots,i_\nu,\ldots i_k}\\ &= [\delta]_{i_\mu,i_\nu}[a]_{j_1,\ldots,j_k}\prod_{q\in[k]}[M(g)]_{i_q,j_q}\\ &= [\delta]_{i_\mu,i_\nu}[a]_{j_1,\ldots,j_k}[M(g)]_{i_\mu,j_\mu}[M(g)]_{i_\nu,j_\nu}\prod_{q\in[k]\setminus\{\mu,\nu\}}[M(g)]_{i_q,j_q}\\ &= \big([\delta]_{i_\mu,i_\nu}[M(g)]_{i_\mu,j_\mu}[M(g)]_{i_\nu,j_\nu}\big)[a]_{j_1,\ldots,j_k}\prod_{q\in[k]\setminus\{\mu,\nu\}}[M(g)]_{i_q,j_q}\\ &= [\delta]_{j_\mu,j_\nu}[a]_{j_1,\ldots,j_k}\prod_{q\in[k]\setminus\{\mu,\nu\}}[M(g)]_{i_q,j_q}\end{aligned}$$

Note that $\big([\delta]_{i_\mu,i_\nu}[M(g)]_{i_\mu,j_\mu}[M(g)]_{i_\nu,j_\nu}\big)$ is the action of g on δ. Thus it equals $[\delta]_{j_\mu j_\nu}$ by Proposition 6, as shown in the last step. Hence:

$$\begin{aligned}
T(g\cdot A,\mu,\nu)(\bar\imath) &= [T(a,\mu,\nu)]_{\{j_1,\ldots,j_k\}\setminus\{j_\mu,j_\nu\}} \prod_{q\in[k]\setminus\{\mu,\nu\}} [M(g)]_{i_q,j_q} \\
&= g\cdot T(A(g^{-1}\cdot\bar\imath),\mu,\nu) \\
&= g\cdot T(A,\mu,\nu)(g^{-1}\cdot\bar\imath) \\
&= (g\cdot T(A,\mu,\nu))(\bar\imath)
\end{aligned}$$

Therefore, since contractions are equivariant to the generators of $G_{N,d}$, it is equivariant to the group.

Next we will prove (38). Let A, B, μ, and ν be defined as above, let $\bar\imath$ be a pixel index of $(\alpha A + \beta B)$, and let $a, b \in \mathcal{T}_{d,k,p}$ be the tensors of A and B at that pixel index. Then:

$$\begin{aligned}
T(\alpha A+\beta B,\mu,\nu)(\bar\imath)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} &= T(\alpha A(\bar\imath)+\beta B(\bar\imath),\mu,\nu)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} \\
&= T(\alpha a+\beta b,\mu,\nu)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} \\
&= \delta_{i_\mu,i_\nu}(\alpha a+\beta b)_{i_1,\ldots,i_k} \\
&= \delta_{i_\mu,i_\nu}(\alpha a)_{i_1,\ldots,i_k} + \delta_{i_\mu,i_\nu}(\beta b)_{i_1,\ldots,i_k} \\
&= \alpha\big(\delta_{i_\mu,i_\nu}a_{i_1,\ldots,i_k}\big) + \beta\big(\delta_{i_\mu,i_\nu}b_{i_1,\ldots,i_k}\big) \\
&= \alpha\cdot T(a,\mu,\nu)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} \\
&\quad + \beta\cdot T(b,\mu,\nu)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} \\
&= \alpha\cdot T(A,\mu,\nu)(\bar\imath)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu} \\
&\quad + \beta\cdot T(B,\mu,\nu)(\bar\imath)_{\{i_1,\ldots,i_k\}\setminus i_\mu,i_\nu}
\end{aligned}$$

Thus we have shown (38).

Now we will prove (39). Let A, S be as described, let $\mu,\nu\in[k]$ distinct, let $\bar\imath$ be a pixel index of A and S, and let $A(\bar\imath) = a$ and $S(\bar\imath) = s$ be the tensors at that pixel index. Then:

$$\begin{aligned}
[T(A\otimes S,\mu,\nu)(\bar\imath)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} &= [T(A(\bar\imath)\otimes S(\bar\imath),\mu,\nu)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} \\
&= [T(a\otimes s,\mu,\nu)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} \\
&= [\delta]_{i_\mu,i_\nu}[a\otimes s]_{i_1,\ldots,i_{k+k'}} \\
&= [\delta]_{i_\mu,i_\nu}[a]_{i_1,\ldots i_k}[s]_{i_{k+1},\ldots,i_{k+k'}} \\
&= [T(a,\mu,\nu)]_{i_1,\ldots,i_k\setminus\{i_\mu,i_\nu\}}[s]_{i_{k+1},\ldots,i_{k+k'}} \\
&= [T(a,\mu,\nu)\otimes s]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} \\
&= [(T(A,\mu,\nu)\otimes S)(\bar\imath)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}}
\end{aligned}$$

This gives us (39). Now suppose instead, $\mu,\nu\in\{k+1,\ldots,k+k'\}$. Skipping a few of the initial steps that are the same as above, we have:

$$\begin{aligned}
[T(A\otimes S,\mu,\nu)(\bar\imath)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} &= [\delta]_{i_\mu,i_\nu}[a]_{i_1,\ldots i_k}[s]_{i_{k+1},\ldots,i_{k+k'}} \\
&= [a]_{i_1,\ldots i_k}[\delta]_{i_\mu,i_\nu}[s]_{i_{k+1},\ldots,i_{k+k'}} \\
&= [a]_{i_1,\ldots i_k}[T(s,\mu-k,\nu-k)]_{i_{k+1},\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} \\
&= [a\otimes T(s,\mu-k,\nu-k)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}} \\
&= [(A\otimes T(S,\mu-k,\nu-k))(\bar\imath)]_{i_1,\ldots,i_{k+k'}\setminus\{i_\mu,i_\nu\}}
\end{aligned}$$

Thus we have our result.

The properties (41) and (42) follow from (39) and (40). Let A, C be as described above, let $\mu, \nu \in [k]$ distinct, and let $\bar{\imath}$ be a pixel index of $A*C$. Then by the previous results and the linearity of contraction we have:

$$\begin{aligned}
T(A*C,\mu,\nu)(\bar{\imath}) &= T((A*C)(\bar{\imath}),\mu,\nu) \\
&= T\left(\sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a})\otimes C(\bar{a}+\bar{m}),\mu,\nu\right) \\
&= \sum_{\bar{a}\in[-m,m]^d} T(A(\bar{\imath}-\bar{a})\otimes C(\bar{a}+\bar{m}),\mu,\nu) \\
&= \sum_{\bar{a}\in[-m,m]^d} T(A(\bar{\imath}-\bar{a}),\mu,\nu)\otimes C(\bar{a}+\bar{m}) \\
&= T(A,\mu,\nu)*C
\end{aligned}$$

Likewise, if $\mu, \nu \in \{k+1,\ldots,k+k'\}$ distinct then:

$$\begin{aligned}
T(A*C,\mu,\nu)(\bar{\imath}) &= \sum_{\bar{a}\in[-m,m]^d} T(A(\bar{\imath}-\bar{a})\otimes C(\bar{a}+\bar{m}),\mu,\nu) \\
&= \sum_{\bar{a}\in[-m,m]^d} A(\bar{\imath}-\bar{a})\otimes T(C(\bar{a}+\bar{m}),\mu-k,\nu-k) \\
&= A*T(C,\mu-k,\nu-k)
\end{aligned}$$

Thus we have our result. □

Next we will state one property of the Levi-Civita contraction, but many properties of regular contractions follow for Levi-Civita Contractions.

Properties (Levi-Civita Contraction)**.** Let $A \in \mathcal{A}_{N,d,k,p}$ for $k \geq d-1, C \in \mathcal{A}_{M,d,k',p'}$, and $\mu_1,\ldots,\mu_{d-1} \in [k]$ distinct. Then the following properties hold:

1. Levi-Civita Contractions are equivariant to $G_{N,d}$:

$$g\cdot T_{LC}(A,\mu_1,\ldots,\mu_{d-1}) = T_{LC}(g\cdot A,\mu_1,\ldots,\mu_{d-1}) \tag{43}$$

Proof. W will prove (43) using the equivariance of the contraction (37). Let A and $\mu_1,\ldots,\mu_\ell$ be as described and let $\bar{\imath}$ be a pixel index. Then

$$\begin{aligned}
(g\cdot T_{LC}(A,\mu_1,\ldots,\mu_{d-1}))(\bar{\imath}) &= g\cdot\big(T_{LC}(A,\mu_1,\ldots,\mu_{d-1})(g^{-1}\cdot\bar{\imath})\big) \\
&= g\cdot\big(T_{LC}(A(g^{-1}\cdot\bar{\imath}),\mu_1,\ldots,\mu_{d-1})\big) \\
&= g\cdot T_M\big(A\big(g^{-1}\cdot\bar{\imath}\big)\otimes\epsilon,(\mu_1,k'+1),\ldots,(\mu_{d-1},k'+d-1)\big) \\
&= T_M\big(g\cdot\big(A\big(g^{-1}\cdot\bar{\imath}\big)\otimes\epsilon\big),(\mu_1,k'+1),\ldots,(\mu_{d-1},k'+d-1)\big) \\
&= T_M\big(g\cdot A\big(g^{-1}\cdot\bar{\imath}\big)\otimes g\cdot\epsilon,(\mu_1,k'+1),\ldots,(\mu_{d-1},k'+d-1)\big) \\
&= T_M((g\cdot A)(\bar{\imath})\otimes\epsilon,(\mu_1,k'+1),\ldots,(\mu_{d-1},k'+d-1)) \\
&= T_{LC}(g\cdot A,\mu_1,\ldots,\mu_{d-1})(\bar{\imath})
\end{aligned}$$

This proof relies on the fact that $g\cdot\epsilon=\epsilon$ given by Proposition 7. Thus we have our result. □

Before proving Proposition 1, we must prove a short lemma about performing a convolution with a filter of size $N+1$ on an image of size N.

Lemma 2. Given $A \in \mathcal{A}_{N,d,k,p}$ a geometric image and $C \in \mathcal{A}_{N+1,d,k',p'}$ a geometric filter where $M = N+1$, there exists $C' \in \mathcal{A}_{N+1,d,k',p'}$ such that $A*C' = A*C$ and $C'(\bar{\imath})$ is the zero $k'_{(p')}$-tensor, for $\bar{\imath} \in [0,N]^d \setminus [0,N-1]^d$. That is, C' is totally defined by N^d pixels, and every pixel with an N in the index is equal to the zero $k'_{(p')}$-tensor.

Proof. Let A and C be defined as above. Thus

$$M = N + 1 \Rightarrow 2m + 1 = N + 1 \Rightarrow 2m = N$$

Consider the convolution definition (11) where we have $A(\bar{\imath} - \bar{a})$ where $\bar{\imath} \in [0, N-1]^d$ and $\bar{a} \in [-m, m]^d$. Since A is on the d-torus, then whenever the ℓ^{th} index of $\bar{a} = -m$ we have:

$$\begin{aligned}(\bar{\imath}_\ell - \bar{a}_\ell) \mod N &= (\bar{\imath}_\ell - (-m)) \mod N \\ &= (\bar{\imath}_\ell + m) \mod 2m \\ &= (\bar{\imath}_\ell + m - 2m) \mod 2m \\ &= (\bar{\imath}_\ell - m) \mod N\end{aligned}$$

Thus, any time there is an index $\bar{a}$ with a value $\pm m$, we have an equivalence class under the torus with all other indices with flipped sign of the m in any combination. If $\{\bar{a}\}$ is this equivalence class, we may group these terms in the convolution sum:

$$\sum_{\bar{a}' \in \{\bar{a}\}} A(\bar{\imath} - \bar{a}') \otimes C(\bar{a}' + \bar{m}) = \sum_{\bar{a}' \in \{\bar{a}\}} A(\bar{\imath} - \bar{a}) \otimes C(\bar{a}' + \bar{m}) = A(\bar{\imath} - \bar{a}) \otimes \left(\sum_{\bar{a}' \in \{\bar{a}\}} C(\bar{a}' + \bar{m}) \right)$$

Thus, we may pick a single pixel of the convolutional filter C, set it equal to $\sum_{\bar{a}' \in \{\bar{a}\}} C(\bar{a}' + \bar{m})$, and set all other pixels of the equivalence class to the zero $k'_{(p')}$-tensor without changing the convolution. We choose the nonzero pixel to be the one whose index has all $-m$ instead of m. Thus we can define the filter C by N^d pixels rather than $(N+1)^d$ pixels, and we have our result. □

Proof of Proposition 1:

Proposition. A function $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k+k',p\,p'}$ is a translation equivariant linear function if and only if it is the convolution with a geometric filter $C \in \mathcal{A}_{M,d,2k+k',p'}$ followed by k contractions. When N is odd, $M = N$, otherwise $M = N + 1$.

Proof. Let $\mathcal{F} = \{f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k+k',p\,p'}\}$ where each function f is linear and equivariant to translations. Let $\mathcal{G} = \{g : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k+k',p\,p'}\}$ where each g is defined as $g(A) = T_M(A * C, (1, k+1), \ldots, (k, 2k))$ for some $C \in \mathcal{A}_{M,d,2k+k',p'}$. If N is odd then $M = N$, otherwise $M = N + 1$. It suffices to show that $\mathcal{F} = \mathcal{G}$.

First we will show that $\mathcal{G} \subseteq \mathcal{F}$. Let $g \in \mathcal{G}$. By properties (33) and (38) both convolutions and contractions are linear. Additionally, by properties (32) and (37) convolutions and contractions are both equivariant to translations. Thus $g \in \mathcal{F}$, so $\mathcal{G} \subseteq \mathcal{F}$.

Now we will show that $\dim(\mathcal{F}) = \dim(\mathcal{G})$. Let $f \in \mathcal{F}$. By Definition 1, $\mathcal{T}_{d,k,p} \cong \left(\mathbb{R}^d\right)^{\otimes k}$ equipped with the group action of $O(d)$. Then by Definition 8, $\mathcal{A}_{N,d,k,p}$ is the space of functions $A : [N]^d \to \mathcal{T}_{d,k,p}$ where $[N]^d$ has the structure of the d-torus. Therefore, $\mathcal{A}_{N,d,k,p} \cong \left(\mathbb{R}^N\right)^{\otimes d} \times \left(\mathbb{R}^d\right)^{\otimes k}$ equipped with the group action of $G_{N,d}$. Thus, $f : \left(\mathbb{R}^N\right)^{\otimes d} \times \left(\mathbb{R}^d\right)^{\otimes k} \to \left(\mathbb{R}^N\right)^{\otimes d} \times \left(\mathbb{R}^d\right)^{\otimes (k+k')}$. Since f is linear, the dimension of the space of functions $\mathcal{F}$ is $N^d d^{k+k'} N^d d^k = N^{2d} d^{2k+k'}$. If this is unclear, consider the fact that the linearity of f means that it has an associated matrix F with that dimension. However, since each f is translation equivariant, the function to each of the N^d pixels in the output must be the same. Thus we actually have that $\dim(\mathcal{F}) = \frac{N^{2d} d^{2k+k'}}{N^d} = N^d d^{2k+k'}$.

Each function $g \in \mathcal{G}$ is defined by the convolution filter C, so $\dim(\mathcal{G}) \leq \dim(\mathcal{A}_{M,d,2k+k',p'}) = \dim(\mathcal{A}_{N,d,2k+k',p'}) = N^d d^{2k+k'}$. The first equality follows from

Lemma 2 in both the even and odd case. The reason this is an inequality rather than an equality is because it is possible that two linearly independent convolution filters result in identical functions g. We will now show that this is not possible.

Suppose $C_1, C_2 \in \mathcal{A}_{M,d,2k+k',+}$ are linearly independent, so for $\alpha, \beta \in \mathbb{R}$ we have that $\alpha C_1 + \beta C_2 = 0$ if and only if $\alpha = \beta = 0$. Now let $g_1, g_2 \in \mathcal{G}$ be defined with convolution filters C_1 and C_2 respectively. Thus it suffices to show that $\alpha g_1 + \beta g_2$ is equal to the function that sends all inputs to the 0 vector, in this case the zero image, if and only if $\alpha = \beta = 0$. Let $A \in \mathcal{A}_{N,d,k,p}$ and by the linearity of contraction (38) and convolution (34) we have:

$$\begin{aligned}\alpha g_1(A) + \beta g_2(A) &= \alpha T_M(A * C_1, (1, k+1), \ldots, (k, 2k)) + \alpha T_M(A * C_2, (1, k+1), \ldots, (k, 2k)) \\ &= T_M(\alpha(A * C_1) + \beta(A * C_2), (1, k+1), \ldots, (k, 2k)) \\ &= T_M(A * (\alpha C_1 + \beta C_2), (1, k+1), \ldots, (k, 2k))\end{aligned}$$

If $\alpha = \beta = 0$ then clearly $\alpha g_1(A) + \beta g_2(A)$ is the zero geometric image for all A.

Now suppose that α and β are not both equal to 0. Then by our linear independence assumption, $\alpha C_1 + \beta C_2$ is not equal to the all zeros filter. Thus there must be at least one component of one pixel that is nonzero. Suppose this is at pixel index $\bar{b} + \bar{m}$ and $(\alpha C_1 + \beta C_2)(\bar{b} + \bar{m}) = c$. Suppose the nonzero component is at index $j_1, \ldots, j_{2k+k'}$. Let $a \in \mathcal{T}_{d,k,p}$ where $[a]_{i_1,\ldots,i_k}$ is nonzero and all other indices are 0. Now suppose $A \in \mathcal{A}_{N,d,k,p}$ such that for pixel index $\bar{\imath}$ of A, $A(\bar{\imath} - \bar{b}) = a$ and all other pixels are the zero tensor. Then:

$$\begin{aligned}(\alpha g_1(A) + \beta g_2(A))(\bar{\imath}) &= T_M(A * (\alpha C_1 + \beta C_2), (1, k+1), \ldots, (k, 2k))(\bar{\imath}) \\ &= T_M((A * (\alpha C_1 + \beta C_2))(\bar{\imath}), (1, k+1), \ldots, (k, 2k)) \\ &= T_M\left(\sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath} - \bar{a}) \otimes (\alpha C_1 + \beta C_2)(\bar{a} + \bar{m}), (1, k+1), \ldots, (k, 2k)\right) \\ &= T_M\big(A(\bar{\imath} - \bar{b}) \otimes (\alpha C_1 + \beta C_2)(\bar{b} + \bar{m}), (1, k+1), \ldots, (k, 2k)\big) \\ &= T_M(a \otimes c, (1, k+1), \ldots, (k, 2k))\end{aligned}$$

Note that the penultimate step removing the sum is because $A(\bar{\imath} - \bar{a}) = 0$ the zero tensor everywhere other than $A(\bar{\imath} - \bar{b})$. Since the only nonzero entry of a is at index $i_1, \ldots, i_k$, then at index $j_{k+1}, \ldots, j_{2k+k'}$ of the resulting tensor we have:

$$[(\alpha g_1(A) + \beta g_2(A))(\bar{\imath})]_{j_{k+1},\ldots,j_{2k+k'}} = [a]_{i_1 \ldots i_k}[c]_{j_1,\ldots,j_{2k+k'}}$$

Since $[a]_{i_1,\ldots,i_k}$ is nonzero and $[c]_{j_1,\ldots,j_{2k+k'}}$ is nonzero, this index is nonzero. Thus the function is not identically 0, so g_1 and g_2 are linearly independent. Therefore, $\dim(\mathcal{G}) = N^d d^{2k+k'}$ and since $\mathcal{G} \subseteq \mathcal{F}$ we have $\mathcal{F} = \mathcal{G}$. □

Proof of Proposition 4

Proposition. Let $\mathcal{F}$ be the set of functions $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$ where each f is a convolution with a B_d-invariant $k'_{(p')}$-tensor filter. Let $\mathcal{G}$ be the set of functions $g : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p''}$ where each g is a convolution with a B_d-invariant $(k'+2)_{(p')}$-tensor filter followed by a contraction. Then $\mathcal{F} \subseteq \mathcal{G}$.

Proof. Let $\mathcal{F}$ and $\mathcal{G}$ be defined as above, let $f \in \mathcal{F}$ with its associated filter $C \in \mathcal{A}_{M,d,k',p'}$, and let $A \in \mathcal{A}_{N,d,k,p}$. Then by Proposition 2, the filter $C' = \left(\frac{1}{d}\right) C \otimes \Delta \in$

$\mathcal{A}_{M,d,k'+2,p'}$ is B_d-invariant. Then by Propositions 40 and 42,

$$\begin{aligned}
f &= A * C \\
&= A * \left(C \otimes \left(\frac{1}{d}\right) d\right) \\
&= A * \left(C \otimes \left(\frac{1}{d}\right) T(\Delta, 1, 2)\right) \\
&= A * T\left(\left(\frac{1}{d}\right) C \otimes \Delta, k'+1, k'+2\right) \\
&= A * T(C', k'+1, k'+2) \\
&= T(A * C', k+k'+1, k+k'+2) \\
&\in \mathcal{G}
\end{aligned}$$

Thus $f \in \mathcal{G}$, so $\mathcal{F} \subseteq \mathcal{G}$. $\square$

Proof of Proposition 5

Proposition. Let $\mathcal{F}$ be the set of functions that preserve parity $f : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p}$ where each f is a convolution with a negative-parity $k'_{(-)}$-tensor filter followed by a Levi-Civita contraction. Let $\mathcal{G}$ be the set of functions that preserve parity $g : \mathcal{A}_{N,d,k,p} \to \mathcal{A}_{N,d,k'',p}$ where each g is a convolution with a positive-parity $(k'+d)_{(+)}$-tensor followed by $d-1$ contractions. Then $\mathcal{F} \subseteq \mathcal{G}$.

Proof. Let $\mathcal{F}$ and $\mathcal{G}$ be as described. Let $A \in \mathcal{A}_{N,d,k,p}, \widetilde{C} \in \mathcal{A}_{M,d,k',-}$, and $\mu_1, \ldots, \mu_{d-1} \in [k+k']$ all distinct. Also let $\bar{\imath}$ be a pixel index and let $E \in \mathcal{A}_{M,d,d,-}$ be the geometric image with the Levi-Civita symbol ϵ in every pixel. Then if $f(A) = T_{LC}\left(A * \widetilde{C}, \mu_1, \ldots, \mu_{d-1}\right)$ we have:

$$\begin{aligned}
T_{LC}\left(A * \widetilde{C}, \mu_1, \ldots, \mu_{d-1}\right)(\bar{\imath}) &= T_{LC}\left(\left(A * \widetilde{C}\right)(\bar{\imath}), \mu_1, \ldots, \mu_{d-1}\right) \\
&= T_M\left(\left(A * \widetilde{C}\right)(\bar{\imath}) \otimes \epsilon, (\mu_1, k+k'+1), \ldots, (\mu_{d-1}, k+k'+d-1)\right) \\
&= T_M\left(\left(\sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes \widetilde{C}(\bar{a}+\bar{m})\right) \otimes \epsilon, (\mu_1, k+k'+1), \ldots\right) \\
&= T_M\left(\sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes \widetilde{C}(\bar{a}+\bar{m}) \otimes \epsilon, (\mu_1, k+k'+1), \ldots\right) \\
&= T_M\left(\sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes \widetilde{C}(\bar{a}+\bar{m}) \otimes E(\bar{a}+\bar{m}), (\mu_1, k+k'+1), \ldots\right) \\
&= T_M\left(\sum_{\bar{a} \in [-m,m]^d} A(\bar{\imath}-\bar{a}) \otimes \left(\widetilde{C} \otimes E\right)(\bar{a}+\bar{m}), (\mu_1, k+k'+1), \ldots\right) \\
&= T_M\left(A * \left(\widetilde{C} \otimes E\right), (\mu_1, k+k'+1), \ldots, (\mu_{d-1}, k+k'+d-1)\right)(\bar{\imath})
\end{aligned}$$

Note that in some of the middle steps we omit some of the multicontraction indices for readability. Now we note that $\widetilde{C} \otimes E$ has tensor order $k'+d$ and parity $-1 * -1 = +1$. Finally, we can show that $\widetilde{C} \otimes E$ is B_d-invariant. For pixel index $\bar{\imath}$ and $g \in B_d$:

$$(g \cdot E)(\bar{\imath}) = g \cdot E(g^{-1} \cdot \bar{\imath}) = g \cdot \epsilon = \epsilon = E(\bar{\imath})$$

which follows from the B_d invariance of the Levi-Civita symbol, Propostion 7. Thus $g \cdot \left(\widetilde{C} \otimes E\right) = g \cdot \widetilde{C} \otimes g \cdot E = \widetilde{C} \otimes E$. Thus $\widetilde{C} \otimes E \in \mathcal{A}_{M,d,k'+d,+}$ is B_d-invariant so $f \in \mathcal{G}$. $\square$

B Related work (mathematical details)

The most common method to design equivariant maps is via group convolution, on the group or on the homogeneous space where the features lie. Regular convolution of a vector field $f : (\mathbb{Z}/N\mathbb{Z})^d \to \mathbb{R}^c$ and a filter $\phi : (\mathbb{Z}/N\mathbb{Z})^d \to \mathbb{R}^c$ is defined as

$$f * \phi(x) = \sum_{y \in (\mathbb{Z}/N\mathbb{Z})^d} \underbrace{\langle f(y), \phi(y-x)\rangle}_{\text{scalar product of vectors}} = \sum_{y \in (\mathbb{Z}/N\mathbb{Z})^d} \sum_{j=1}^{c} \underbrace{f^j(y)\phi^j(y-x)}_{\in \mathbb{R}} \tag{44}$$

Our generalization of convolution replaces this scalar product of vectors by the outer product of tensors. Probably the most related work is by Brandstetter et al. [4] who replace it by the geometric product of multivector inputs and multivector filters of a Clifford Algebra.

B.1 Clifford Convolution

This paper [4] deals with multivector fields, i.e.: vector fields $f : \mathbb{Z}^2 \to (Cl_{p,q}(\mathbb{R}))^c$. The real Clifford Algebra $Cl_{p,q}(\mathbb{R})$ is an associative algebra generated by $p+q=d$ orthonormal basis elements: $e_1, \ldots, e_{p+q} \in \mathbb{R}^d$ with the relations:

$$e_i \otimes e_i = +1 \qquad (i \leq p), \tag{45}$$

$$e_j \otimes e_j = -1 \quad (p < j \leq n), \tag{46}$$

$$e_i \otimes e_j = -e_j \otimes e_i \qquad (i \neq j). \tag{47}$$

For instance, $Cl_{2,0}(\mathbb{R})$ has the basis $\{1, e_1, e_2, e_1 \otimes e_2\}$ and is isomorphic to the quaternions $\mathbb{H}$.

The Clifford convolution replaces the elementwise product of scalars of the usual convolution of (44) by the geometric product of multivectors in the Clifford Algebra:

$$f * \phi(x) = \sum_{y \in (\mathbb{Z}/N\mathbb{Z})^d} \sum_{j=1}^{c} \underbrace{f^j(y) \otimes \phi^j(y-x)}_{\in Cl_{p,q}(\mathbb{R})}, \tag{48}$$

where $f : \mathbb{Z}^2 \to (Cl_{p,q}(\mathbb{R}))^c$ and $\phi : \mathbb{Z}^2 \to (Cl_{p,q}(\mathbb{R}))^c$

The Clifford Algebra $Cl_{p,q}(\mathbb{R})$ is a quotient of the tensor algebra

$$T(\mathbb{R}^d) = \bigoplus_{k \geq 0} \underbrace{\mathbb{R}^d \otimes \ldots \otimes \mathbb{R}^d}_{k \text{ times}} = \bigoplus_{k \geq 0} (\mathbb{R}^d)^{\otimes k}, \tag{49}$$

by the two-side ideal $\langle \{v \otimes v - Q(v) : v \in \mathbb{R}^d\} \rangle$, where the quadratic form Q is defined by $Q(e_i) = +1$,if $i \leq p$, and $Q(e_j) = -1$, else $p < j \leq n$. Our geometric images are functions $A : (\mathbb{Z}/N\mathbb{Z})^d \to \mathcal{T}_{d,k,p}$, where $\mathcal{T}_{d,k,p} = (\mathbb{R}^d)^{\otimes k} \subset T(\mathbb{R}^d)$. They can be related with the Clifford framework by seeing them as N-periodic functions from $\mathbb{Z}^d$ whose image is projected via the quotient map on the Clifford Algebra. This projection can be seen as a contraction of tensors.

The Clifford convolution is not equivariant under multivector rotations or reflections. But the authors derive a constraint on the filters for $d = 2$ which allows to build generalized Clifford convolutions which are equivariant with respect to rotations or reflections of the multivectors. That is, they prove equivariance of a Clifford layer under orthogonal transformations if the filters satisfies the constraint: $\phi^i(Rx) = R\phi^i(x)$.

Part of our work can be studied under the unified framework for group equivariant networks on homogeneous spaces derived from a Fourier perspective proposed in [49].

B.2 Unified Fourier Framework

In [49], they consider general tensor-valued feature fields, before and after a convolution. Their fields are functions $f : G/H \to V$ over the homogeneous space G/H taking values in the vector space V and their filters are kernels $\kappa : G \to Hom(V, V')$. Essentially, their convolution replaces the scalar product of vectors of traditional convolution by appliying an homomorphism. In particular, if G is a finite group and $H = \{0\}$, they define convolution as

$$\kappa * f(x) = \frac{1}{|G|} \sum_{y \in G} \underbrace{\kappa(x^{-1}y) f(y)}_{\in V'} . \tag{50}$$

They give a complete characterization of the space of kernels for equivariant convolutions. In our framework, the group is $\mathbb{Z}/N\mathbb{Z}$ and the kernel is an outer product by a filter C: $\kappa(g)A(g) = A(g) \otimes C(g)$. Note that $\mathbb{Z}/N\mathbb{Z}$ is neither an homogeneous space of $O(d)$ nor of B^d

We can analize our problem from an spectral perspective, in particular we can describe all linear equivariant using representation theory, using similar tools as in the proof of the "only if" part of Theorem 1 in [29]. This theorem states that convolutional structure is a sufficient and a necessary condition for equivariance to the action of a compact group. Some useful references about group representation theory are [18], a classical book about the theory of abstract harmonic analysis and [5], about the particular applications of it.

B.3 Linear equivariant maps

In this work we define an action over tensor images of $O(d)$, by rotation of tensors in each pixel; of B^d by rotating the grid of pixels and each tensor in the pixel; and of $(\mathbb{Z}/N\mathbb{Z})^d$ by translation of the grid of pixels. The action of each one of these groups G over $\mathcal{T}_{d,k,p}$

$$\Phi_{d,k,p} : G \to GL_{con}(\mathcal{T}_{d,k,p}), \tag{51}$$

can be decomposed into irreducible representations of G:

$$\Phi_{d,k,p} \equiv \bigoplus_{\pi \in \hat{G}} m_{d,k,p}(\pi)\pi. \tag{52}$$

That is, there is a basis of the Hilbert space $\mathcal{T}_{d,k,p}$ in which the action of G is defined via a linear sparse map. In the case of G finite, for all $g \in G$ there is a matrix P splitting the representation in the Hilbert space into its irreducible components

$$P^{-1}\Phi_{d,k,p}(g)P = \bigoplus_{\pi \in \hat{G}} m_{d,k,p}(\pi)\pi(g) \tag{53}$$

Consider now linear maps between Tensor images:

$$\mathcal{C} : \mathcal{T}_{d,k,p} \to \mathcal{T}_{d',k',p'} \tag{54}$$

Linear equivariant maps satisfy that $\mathcal{C} \circ \Phi_{d,k,p} = \Phi_{d',k',p'} \circ \mathcal{C}$. That is, if $\tilde{\mathcal{C}}$ is the representation of $\mathcal{C}$ in the above basis,

$$\tilde{\mathcal{C}} \circ \bigoplus_{\pi \in G} m_{d,k,p}(\pi)\pi = \bigoplus_{\pi \in G} m_{d',k',p'}(\pi)\pi \circ \tilde{\mathcal{C}}. \tag{55}$$

By Schur's Lemma, this implies that $\mathcal{C} \equiv \bigoplus_{\pi \in G} m_{d,k,p}(\pi) Id_{d_\pi}$.

The power of representation theory is not limited to compact groups. Mackey machinery allow us to study for instance semidirect products of compact groups and other groups, and in general to relate the representations of a normal subgroup with the ones of the hole group. This is the spirit of [8], which makes extensive use of the induced representation theory. An introduction to this topic can be found in Chapter 7 in [18].

B.4 Steerable CNNs

This paper [8] deals exclusively with signals $f : \mathbb{Z}^2 \to \mathbb{R}^k$. They consider the action of $p4m$ on $\mathbb{Z}^2$ by translations, rotations by 90 degrees around any point, and reflections. This group is a semidirect product of $\mathbb{Z}^2$ and $H = D^4$ (also called B^2). Every $x \in p4m$ can be written as $x = tr$, for $t \in \mathbb{Z}^2$ and $r \in D^4$. They show that equivariant maps with respect to representations ρ and ρ' of rotations and reflections H lead to equivariant maps with respect to certain representations of G π and π'. This means that if we find a linear map $\phi : f \mapsto \phi f$ such that $\phi\rho(h)f = \rho'(h)\phi f$ for all $h \in H$, then for the representation of G π' defined by

$$\pi'(tr)f(y) = \rho(r)[f((tr)^{-1}y)], \quad tr \in G, y \in \mathbb{Z}^2, \tag{56}$$

we automatically have that $\phi\pi(g)f = \pi'(g)\phi f$ for all $g \in G$. This is the representation of G induced by the representation ρ of H

Note that, the similarity between the definition of the action of B^d on tensor images and equation (56). The convolution with a symmetric filter produces easily an equivariant map with respect to the action of the semidirect product of $\mathbb{Z}^d$ and B^d on the tensor images.

B.5 Equivariant dynamics

In the recent papers of Rose Yu et al. [46], they study approximately equivariant networks which are biased towards preserving symmetry but are not strictly constrained to do so. They define a relaxed group convolution which is approximately equivariant in the sense that

$$\|\rho_X(g)f *_G \Psi(x) - f *_G \Psi(\rho_Y(y)x\| < \epsilon. \tag{57}$$

They use a classical convolution but with different kernels for different group elements.