The Journal of Open Source Software

# GeometricConvolutions: E(d)-Equivariant CNN for Tensor Images

**Wilson G. Gregory** [1¶] **and Kaze W. K. Wong**[1]

**1** Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA ¶ Corresponding author

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Open Journals

**Reviewers:**

- @openjournals

**Submitted:** 01 January 1970
**Published:** unpublished



## Summary

Many data sets encountered in machine learning exhibit symmetries that can be exploited to improve performance, a technique known as equivariant machine learning. The classical example is image translation equivariance that is respected by convolutional neural networks (LeCun et al., 1989). For data sets in the physical sciences and other areas, we would also like equivariance to rotations and reflections. This Python package implements a convolutional neural network that is equivariant to translations, rotations of 90 degrees, and reflections. We implement this by *geometric convolutions* (Gregory et al., 2024) which use tensor products and tensor contractions. This additionally enables us to perform functions on geometric images, or images where each pixel is a higher order tensor. These images appear as discretizations of fields in physics, such as velocity fields, vorticity fields, magnetic fields, polarization fields, and so on.

This package includes basic functionality to create, manipulate, and plot geometric images using JAX (Bradbury et al., 2018). It also provides equivariant neural network layers such as convolutions, activation functions, group norms, and others using the Equinox framework (Kidger & Garcia, 2021). These layers ingest a special data structure, the `MultiImage`, that allows combining geometric images or any tensor order or parity into a single model. Finally, the package provides full-fledged versions of popular models such as the UNet or ResNet to allow researchers to quickly train and test on their own data sets with standard tools in the JAX ecosystem.

## Statement of need

The geometric convolutions introduced in (Gregory et al., 2024) are defined on geometric images–an image where every pixel is a tensor. If $A$ is a geometric image of tensor order $k$ and $C$ is a geometric image of tensor order $k'$, then value of $A$ convolved with $C$ at pixel $\bar{\imath}$ is given by:

$$(A * C)(\bar{\imath}) = \sum_{\bar{a}} A(\bar{\imath} - \bar{a}) \otimes C(\bar{a}) \ ,$$

where the sum is over all pixels $\bar{a}$ of $C$, and $\bar{\imath} - \bar{a}$ is the translation of $\bar{\imath}$ by $\bar{a}$. The result is a geometric image of tensor order $k + k'$. To produce geometric images of smaller tensor order, the tensor contraction can be applied to each pixel. Convolution and contraction are combined into a single operation to form linear layers. By restricting the convolution filters $C$ to rotation and reflection invariant filters, we can create linear layers which are rotation-, reflection-, and translation-equivariant.

The space of equivariant machine learning software is largely still in its infancy, and this is currently the only package implementing geometric convolutions. However, there are alternative methods for solving $O(d)$-equivariant image problems. One such package is escnn which uses Steerable CNNs (Cohen & Welling, 2016; **?**). Steerable CNNs use irreducible representations to derive a basis for $O(d)$-equivariant layers, but it is not straightforward to apply on higher order tensor images. The escnn package is built with pytorch, although there is a JAX port escnn_jax.

Another alternative method is are those based on Clifford Algebras, in particular (Brandstetter et al., 2023). This method has been implemented in the Clifford Layers package. Like escnn, this method is also built with pytorch rather than JAX. Additionally, Clifford based methods can process vectors and pseudovectors, but cannot handle higher order tensors.

Implementing our library in JAX allows us to easily build and optimize machine learning models with Equinox and Optax (DeepMind et al., 2020). For equivariance researchers, we provide all the common operations on geometric images such as addition, scaling, convolution, contraction, transposition, norms plus visualization methods. For practitioners, we provide both equivariant layers for building models, and full fledged model implementations such as the UNet, ResNet, and Dilated ResNet.

# References


Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Brandstetter, J., Berg, R. van den, Welling, M., & Gupta, J. K. (2023). *Clifford neural layers for PDE modeling*. https://arxiv.org/abs/2209.04934

Cohen, T. S., & Welling, M. (2016). *Steerable CNNs*. https://arxiv.org/abs/1612.08498

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., … Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

Gregory, W. G., Hogg, D. W., Blum-Smith, B., Arias, M. T., Wong, K. W. K., & Villar, S. (2024). *Equivariant geometric convolutions for emulation of dynamical systems*. https://arxiv.org/abs/2305.12585

Kidger, P., & Garcia, C. (2021). Equinox: Neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming Workshop at Neural Information Processing Systems 2021*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.