The Journal of Open Source Software

PyWavelets: A Python package for wavelet analysis

Gregory R. Lee[1, 2], **Ralf Gommers**[3, 4], **Filip Waselewski**[6], **Kai Wohlfahrt**[5], **and Aaron O'Leary**[6]

1 Department of Radiology, Cincinnati Children's Hospital Medical Center, Cincinnati, OH, USA **2** Department of Radiology, University of Cincinnati School of Medicine, Cincinnati, OH, USA **3** Scion, 49 Sala Street, Private Bag 3020, Rotorua 3046, New Zealand **4** FPInnovations, 2665 East Mall, Vancouver, BC V6T 1Z4, Canada **5** Department of Biochemistry, University of Cambridge, Old Addenbrookes Site, 80 Tennis Court Road, Cambridge, CB2 1GA, United Kingdom **6** None

DOI: 10.21105/joss.01237

Software

- Review
- Repository
- Archive

Submitted: 18 December 2018
Published: 12 April 2019

Summary

Wavelets are a popular tool for computational harmonic analysis. They provide localization in both the temporal (or spatial) domain as well as in the frequency domain (Daubechies, 1992). A prominent feature is the ability to perform a multiresolution analysis (S. Mallat, 2008). The wavelet transform of natural signals and images tends to have most of its energy concentrated in a small fraction of the coefficients. This sparse representation property is key to the good performance of wavelets in applications such as data compression and denoising. For example, the wavelet transform is a key component of the JPEG 2000 image compression standard.

`PyWavelets` is a Python package implementing a number of n-dimensional discrete wavelet transforms as well as the 1D continuous wavelet transform. A wide variety of predefined wavelets are provided and it is possible for users to specify custom wavelet filter banks. All discrete wavelet transforms are implemented by convolution with finite impulse response filters. The required up/downsampling convolutions are implemented in C for good performance. Cython (Behnel et al., 2011) is used to wrap the C code and implement axis-specific 1D transformations based on the low-level C routines. All multi-dimensional transforms are implemented in Python via separable application of the 1D transforms. The API for `PyWavelets` was designed to be similar to Matlab's wavelet toolbox and functions such as the 1D, 2D and 3D discrete wavelet transforms are tested for accuracy vs. their Matlab counterparts. PyWavelets has additional functionality not common in other wavelet toolboxes such as support for dimension $n > 3$ and support for both real and complex-valued data in either single or double precision. It is also possible to transform only a subset of axes and to vary the wavelet and signal boundary extension mode on a per-axis basis.

`PyWavelets` was designed for use by scientists working within a range of applications including time-series analysis, signal processing, image processing and medical imaging. It has already been adopted as a required or optional dependency by a number of other software projects. For example, it has enabled wavelet-based image denoising in scikit-image (Walt et al., 2014). The Operator Discretization Library (ODL) (Adler et al., 2018) uses PyWavelets to enable wavelet-based regularization in iterative inverse problems such as computed tomography image reconstruction. Another related package which is independent of `PyWavelets` is Kymatio, which implements the wavelet scattering transform in 1D-3D (Andreux et al., 2018). The current implementation in Kymatio uses non-separable 2D and 3D wavelets defined in the frequency domain and is well suited to signal classification tasks, but does not have a simple inverse transform like the standard discrete wavelet transform.

A number of common 1D demo signals used in the literature and in the manuscript by Stephan Mallat (2008) are provided for use in teaching and for purposes of reproducible research.

Acknowledgements

We would to acknowledge the various PyWavelets contributors for their contributions to the library, and specifically Holger Nahrstaedt for contributing a continuous wavelet transform.

References

Adler, J., Kohr, H., Ringh, A., Moosmann, J., Banert, S., Ehrhardt, M. J., Lee, G. R., et al. (2018, September). Odlgroup/odl: ODL 0.7.0. doi:10.5281/zenodo.1442734

Andreux, M., Angles, T., Exarchakis, G., Leonarduzzi, R., Rochette, G., Thiry, L., Zarka, J., et al. (2018). Kymatio: Scattering transforms in python. *CoRR*, *abs/1812.11214*. Retrieved from http://arxiv.org/abs/1812.11214

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science Engineering*, *13*(2), 31–39. doi:10.1109/MCSE.2010.118

Daubechies, I. (1992). *Ten lectures on wavelets*. Philadelphia, PA, USA: Society for Industrial; Applied Mathematics. doi:10.1137/1.9781611970104

Mallat, S. (2008). *A wavelet tour of signal processing: The sparse way*. Academic press.

Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., et al. (2014). Scikit-image: Image processing in Python. *PeerJ*, *2*, e453. doi:10.7717/peerj.453