Journal of Machine Learning Research 21 (2020) 1-6 Submitted 1/19; Revised 10/19; Published 1/20

Kymatio: Scattering Transforms in Python

Mathieu Andreux MATHIEU.ANDREUX@OWKIN.COM
Tomás Angles TOMAS.ANGLES@ENS.FR
Georgios Exarchakis GEORGIOS.EXARCHAKIS@ENS.FR
Roberto Leonarduzzi ROBERTO.LEONARDUZZI@ENS-LYON.FR
Gaspar Rochette GASPAR.ROCHETTE@ENS.FR
Louis Thiry LOUIS.THIRY@OUTLOOK.FR
John Zarka JOHNZARKA@GMAIL.COM
École normale supérieure, CNRS, PSL Research University, 45, rue d'Ulm, 75005 Paris, France

Stéphane Mallat STEPHANE.MALLAT@ENS.FR
École normale supérieure, CNRS, PSL Research University, 45, rue d'Ulm, 75005 Paris, France
Collège de France, 11, place Marcelin-Berthelot 75231 Paris, France
Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

Joakim Andén JANDEN@FLATIRONINSTITUTE.ORG
Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

Eugene Belilovsky EUGENE.BELILOVSKY@UMONTREAL.CA
Mila, Université de Montréal, 6666 St Urbain Street, Montreal, Quebec H2S 3H1, Canada

Joan Bruna BRUNA@CIMS.NYU.EDU
Vincent Lostanlen VL1019@NYU.EDU
New York University, 70 Washington Square South, New York, NY 10012, USA

Muawiz Chaudhary CHAUDHM@WWU.EDU
Western Washington University, 516 High Street, Bellingham, WA 98225, USA

Matthew J. Hirn MHIRN@MSU.EDU
Michigan State University, 426 Auditorium Road East Lansing, MI 48824, USA

Edouard Oyallon EDOUARD.OYALLON@LIP6.FR
CNRS, LIP6, Sorbonne University, 4 place Jussieu, 75252 Paris, France

Sixin Zhang SIXIN.ZHANG@PKU.EDU.CN
Peking University, No. 5 Yiheyuan Road, Haidian District, Beijing 100871, China

Carmine Cella CARMINE.CELLA@BERKELEY.EDU
University of California, Berkeley, 101 Sproul Hall, Berkeley, CA 94720, USA

Michael Eickenberg MEICKENBERG@FLATIRONINSTITUTE.ORG
Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

Editor: Balazs Kegl

Abstract

The wavelet scattering transform is an invariant and stable signal representation suitable for many signal processing and machine learning applications. We present the *Kymatio* software package, an easy-to-use, high-performance Python implementation of the scattering transform in 1D, 2D, and 3D that is compatible with modern deep learning frameworks, including *PyTorch* and *TensorFlow/Keras*. The transforms are implemented on both CPUs

and GPUs, the latter offering a significant speedup over the former. The package also has a small memory footprint. Source code, documentation, and examples are available under a BSD license at `https://www.kymat.io`.

Keywords: Scattering Transform; GPUs; Wavelets; Convolutional Networks; Invariance

1. Introduction

Many classification and regression tasks have a degree of invariance to translations and deformations, such as those relating to images, audio recordings, and electronic densities. The scattering transform was introduced in Mallat (2012) to build a signal representation that is invariant to such transformations while preserving as much as possible the information relevant to the task at hand. It is defined as a convolutional network whose filters are fixed to be wavelet and lowpass averaging filters coupled with modulus nonlinearities. It has many favorable theoretical properties (Mallat, 2012; Bruna et al., 2015; Waldspurger, 2017) and enjoys considerable success as a powerful tool in modern signal processing (Adel et al., 2017; Bruna and Mallat, 2013; Andén and Mallat, 2014; Chudáček et al., 2014; Sifre and Mallat, 2013; Eickenberg et al., 2017). It is also effective in combination with modern representation learning approaches (Oyallon et al., 2018; Sainath et al., 2014; Zeghidour et al., 2016).

This article presents *Kymatio*, a scattering transform implementation that is user-friendly, well-documented, fast, and compatible with existing automatic differentiation libraries. It brings together transforms in 1D, 2D, and 3D under a *unified* application programming interface (API). The scattering network is also traversed depth-first to reduce memory requirements, enabling efficient processing in limited-memory environments, such as GPUs.

2. Implementing the Scattering Transform

Definition We consider signals defined on a grid of size $N_1 \times \cdots \times N_d$ for $d = 1, 2, 3$. Given two signals $x[n]$ and $y[n]$ on this grid, we denote their periodic convolution by $x \circledast y[n]$. The second-order scattering transform is defined using two wavelet filter banks $\{\psi^{(1)}_{\lambda_1}[n]\}_{\lambda_1 \in \Lambda_1}$ and $\{\psi^{(2)}_{\lambda_2}[n]\}_{\lambda_2 \in \Lambda_2}$, where λ_1 and λ_2 are frequency indices in the sets Λ_1 and Λ_2. It also includes a lowpass filter $\phi_J[n]$, where the integer $J > 0$ specifies the averaging scale 2^J of the filter. Together with a non-linearity $\rho(t)$, these filters define the scattering transform.

The zeroth-order scattering coefficient $S_0x[n]$ is the local average given by $S_0x[n] = x \circledast \phi_J[n]$. Convolving $x[n]$ with the first-order wavelet filter bank $\{\psi^{(1)}_{\lambda_1}[n]\}_{\lambda_1 \in \Lambda_1}$, applying $\rho(t)$, and convolving with $\phi_J[n]$, we obtain the first-order scattering coefficients

$$S_1x[n, \lambda_1] = \rho\left(x \circledast \psi^{(1)}_{\lambda_1}\right) \circledast \phi_J[n], \qquad \lambda_1 \in \Lambda_1. \tag{1}$$

The modulus of the first wavelet transform acts as a demodulation, shifting its energy to the low frequencies. However, only some of these frequencies are covered by the low-pass filter ϕ_J. We recover the remaining frequencies by decomposing $\rho(x \circledast \psi^{(1)}_{\lambda_1}[n])$ using the second filter bank, but this is done only for a subset $\Lambda_2(\lambda_1)$ of Λ_2 since $\rho(x \circledast \psi^{(1)}_{\lambda_1}[n])$ is a low-frequency signal. Typically, we have $\Lambda_2(\lambda_1) = \{\lambda \in \Lambda_2, |\lambda| > |\lambda_1|\}$. The result is then passed through $\rho(t)$ and averaged, yielding the second-order coefficients

$$S_2x[n, \lambda_1, \lambda_2] = \rho\left(\rho\left(x \circledast \psi^{(1)}_{\lambda_1}\right) \circledast \psi^{(2)}_{\lambda_2}\right) \circledast \phi_J[n], \qquad \lambda_1 \in \Lambda_1, \lambda_2 \in \Lambda_2(\lambda_1). \tag{2}$$

	DIMENSION	GPU	DIFF.	CORE DEVS.	LICENSE	LANGUAGE
ScatNet	1D, 2D			5	Apache 2.0	MATLAB
ScatNetLight	2D			2	GPLv2	MATLAB
PyScatWave	2D	✓		3	BSD-3	Python
Scattering.m	1D			1	GPLv3	MATLAB
PyScatHarm	3D	✓		1	BSD-3	Python
Wavelet Toolbox	1D			N/A	Proprietary	MATLAB
Kymatio	1D, 2D, 3D	✓	✓	15	BSD-3	Python

Table 1: Comparison to existing scattering transform packages.

The energy of higher-order scattering coefficients is typically small and does not greatly influence results (Waldspurger, 2017; Bruna and Mallat, 2013; Andén and Mallat, 2014). On the other hand they can be computationally intensive. We have thus chosen to restrict our scope to second-order coefficients, which is what is used in most works.

Implementation Signals obtained by filtering and applying $\rho(t)$ are low-frequency, so intermediate results are downsampled to reduce computational load as in Andén et al. (2014). In 1D and 2D, we use Morlet wavelets which are close to analytic (*i.e.*, complex-valued with low energy in the negative frequencies) and the non-linearity is the complex modulus $\rho(t) = |t|$ for $t \in \mathbb{C}$ (Andén and Mallat, 2014; Bruna and Mallat, 2013). The 3D transform is calculated using solid harmonic wavelets $\psi_{\lambda_1} = \psi_{j,\ell,m}$, where j indexes the scale, and ℓ, m are the azimuthal and magnetic quantum numbers. In this case the non-linearity $\rho : \mathbb{C}^{2\ell+1} \to \mathbb{R}$ is defined, with a slight abuse of notation, as $\rho(x \circledast \psi_{j,\ell}) = \sqrt{\sum_m |x \circledast \psi_{j,\ell,m}|^2}$ (Eickenberg et al., 2017). Following Oyallon et al. (2018), we compute the scattering transform in a depth-first manner, reducing the number of intermediate signals stored at a given time. Since convolutions are all periodic, they may be efficiently calculated using fast Fourier transforms.

3. Project vision

Code quality Adopting the philosophy of *scikit-learn* (Pedregosa et al., 2011), the goal of *Kymatio* is not to maximize the number of features, but to provide a stable and easy-to-use framework. To this end, we make heavy use of unit tests, minimize the number of dependencies, and strive for intuitive interfaces inspired by modern deep learning paradigms. *Kymatio* also provides an extensive user guide, including an API reference, a tutorial, installation instructions, and easy-to-understand examples, several of which feature real-world applications.

Community and bug tracking *Kymatio* is free and open-source software with a 3-clause BSD license. The members of its core development team all have experience implementing scattering transforms in other packages. A key goal of *Kymatio* is to combine these efforts and foster a community effort in order to produce high-quality software and maintain a critical mass of contributors for its maintenance. The package was released publicly on GitHub November 17th, 2018. The main communication channel is the GitHub page for questions, bug reports, and feature requests. There is also a dedicated Slack channel.

Relation to previous software Aside from the emphasis on code quality and usability, *Kymatio* provides several improvements over previous scattering implementations:

- *Python* is the *de facto* standard for data science software, but most existing scattering packages are implemented in MATLAB. In contrast, *Kymatio* provides a completely Pythonic implementation, enabling integration with the scientific Python ecosystem.
- *GPU compatibility* is critical to many data science workloads. *Kymatio* offers an easy-to-use GPU implementation for scattering transforms in 1D, 2D, and 3D.
- *Frontends* are provided for many frameworks, including *NumPy*, *scikit-learn*, *PyTorch*, and *TensorFlow/Keras*, allowing for seamless integrating scattering transforms in a variety of pipelines. In particular, the *PyTorch*, and *TensorFlow/Keras* frontends allow for inclusion into many deep learning workflows.
- *Differentiability* of the scattering transform simplifies applications in reconstruction and generative modeling, among others.

Table 1 provides a detailed comparison of existing implementations: ScatNet (Andén et al., 2014), ScatNetLight (Oyallon and Mallat, 2015), PyScatWave (Oyallon et al., 2018), Scattering.m (Lostanlen and Mallat, 2015), PyScatHarm (Eickenberg et al., 2018), and the scattering transform implemented in the MATLAB Wavelet Toolbox.

4. User Interface and Documentation

Interface The interface is designed to be flexible and consistent across inputs and frontends. Let us consider the *PyTorch* frontend. We first create a scattering object by specifying the averaging scale J and the input signal shape.

```
from kymatio.torch import Scattering1D, Scattering2D, HarmonicScattering3D

S = Scattering1D(J, shape=(length,))
S = Scattering2D(J, shape=(height, width))
S = HarmonicScattering3D(J, shape=(height, width, depth))
```

The resulting object `S` acts like a `nn.Module` object in *PyTorch*. The scattering transform `S` is applied through calls of the form

```
x = torch.randn((28, 28))
output = S(x)
```

Switching from GPU or CPU functionality also follows the API of `nn.module`.

```
S.cuda()  # Run on GPU
S.cpu()   # Run on CPU
```

Documentation and examples Several examples are provided with the code, illustrating the power of *Kymatio*. These include image reconstruction and generation from scattering (Angles and Mallat, 2018), hybrid scattering and CNN training on CIFAR and MNIST (Oyallon et al., 2018), regression of molecular properties on QM7/QM9 using solid harmonic scattering (Eickenberg et al., 2017), and classifying recordings of spoken digits.

5. Conclusion

Kymatio provides a well documented, user-friendly, and fast implementation for the scattering transform. It can be used with the *PyTorch* and *TensorFlow/Keras* deep learning frameworks and supports a variety of applications that have been previously inaccessible to non-experts including hybrid deep learning, generative modeling, and 3D chemistry applications. Future work includes further optimization for speed, flexibility, and backend support.

Acknowledgments

We thank Laurent Sifre, Sergey Zagoruyko and Gabriel Huang for their helpful comments. The project was supported by ERC InvariantClass 320959. EB is funded by a Google Focused Research Award and IVADO. MJH is partially supported by Alfred P. Sloan Fellowship #FG-2016-6607, DARPA Young Faculty Award #D16AP00117, and NSF grant #1620216. The Flatiron Institute is a division of the Simons Foundation.

References

T. Adel, T. Cohen, M. Caan, M. Welling, et al. 3D scattering transforms for disease classification in neuroimaging. *NeuroImage: Clinical*, 14:506–517, 2017. doi: 10.1016/j.nicl.2017.02.004.

J. Andén et al. Scatnet. *Computer Software*, 2014. URL `http://www.di.ens.fr/data/software/scatnet`.

J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Trans. Signal Process.*, 62(16): 4114–4128, Aug 2014. doi: 10.1109/TSP.2014.2326991.

T. Angles and S. Mallat. Generative networks as inverse problems with scattering transforms. In *Proc. ICLR*, 2018.

J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, 2013. doi: 10.1109/TPAMI.2012.230.

J. Bruna, S. Mallat, E. Bacry, and J.-F. Muzy. Intermittent process analysis with scattering moments. *Ann. Statist.*, 43(1):323–351, 02 2015. doi: 10.1214/14-AOS1276.

V. Chudáček et al. Scattering transform for intrapartum fetal heart rate variability fractal analysis: A case-control study. *IEEE Trans. Biomed. Eng.*, 61(4):1100–1108, 2014. doi: 10.1109/TBME.2013.2294324.

M. Eickenberg et al. Solid harmonic wavelet scattering: Predicting quantum molecular energy from invariant descriptors of 3D electronic densities. In *Proc. NIPS*, pages 6540–6549, 2017.

M. Eickenberg et al. Solid harmonic wavelet scattering for predictions of molecule properties. *The Journal of Chemical Physics*, 148(24):241732, 2018. doi: 10.1063/1.5023798.

V. Lostanlen and S. Mallat. Wavelet scattering on the pitch spiral. In *Proc. DAFx*, 2015.

S. Mallat. Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398, 2012. doi: 10.1002/cpa.21413.

E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *Proc. CVPR*, June 2015.

E. Oyallon et al. Scattering networks for hybrid representation learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(9):2208–2221, 2018. doi: 10.1109/TPAMI.2018.2855738.

F. Pedregosa et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12(Oct): 2825–2830, 2011.

T. N. Sainath et al. Deep scattering spectra with deep neural networks for LVCSR tasks. In *Proc. Interspeech*, 2014.

L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proc. CVPR*, 2013. doi: 10.1109/CVPR.2013.163.

I. Waldspurger. Exponential decay of scattering coefficients. In *Proc. SampTA*, pages 143–146, 2017. doi: 10.1109/SAMPTA.2017.8024473.

N. Zeghidour et al. A deep scattering spectrum–deep siamese network pipeline for unsupervised acoustic modeling. In *Proc. ICASSP*, pages 4965–4969. IEEE, 2016. doi: 10.1109/ICASSP.2016.7472622.