xr_fresh: Automated Time Series Feature Extraction for Remote Sensing & Gridded Data

21 May 2025

Abstract

xr_fresh is a Python library for automated feature extraction from gridded time series data, such as satellite imagery, climate model outputs, and sensor arrays. Building on the methodology of tsfresh, xr_fresh extends this approach to pixel-level temporal sequences common in observational data such as from earth observation or repeat photography data. It computes a comprehensive set of statistical, trend, and distribution-based features for each pixel, enabling scalable preprocessing for classical machine learning. The library is optimized for large-scale applications through parallelized computation using xarray, Dask, Ray, and JAX. It also includes advanced interpolation techniques for handling missing data and GPU-accelerated kernel PCA for dimensionality reduction.

Statement of need

Gridded time series data from satellites, climate models, camera feeds, and sensors contain rich temporal information for applications like crop type classification and yields, anomaly detection, robotics, quality control, environmental monitoring, and natural resource management (Delince et al. 2017; Mumuni and Mumuni 2024; Hufkens et al. 2019; Michael L. Mann and Warner 2017; Michael L. Mann, Warner, and Malik 2019). Efficiently extracting relevant time series features at scale remains challenging, necessitating automation (Faouzi 2022; Li, Kang, and Li 2020). Inspired by `tsfresh`, we introduce `xr_fresh`, tailored specifically for gridded time series by automating the extraction of time series features on a pixel-by-pixel basis (Christ et al. 2018).

Currently, there is no method to rapidly extract a comprehensive set of features from gridded time series data, such as those derived from remote sensing imagery. Existing packages like `tsfresh` are not optimized for the unique structure of gridded time series data and take 160 times longer to process. This limitation hinders the ability to efficiently analyze and model these datasets, particularly in the context of remote sensing applications where large volumes of data are generated.

To address this gap, `xr_fresh` automates the extraction of salient temporal and statistical features from each pixel time series. Using automated feature extraction, `xr_fresh` reduces manual intervention and improves reproducibility in remote sensing workflows.

Problems and Background

An image time series can be represented as a three-dimensional array with spatial dimensions x and y, and temporal dimension z. Each pixel at location (x_i, y_j) holds a time series:

$$\mathcal{D} = \{X_{i,j} \in \mathbb{R}^T \mid i = 1, \dots, H; j = 1, \dots, W\}$$

where H and W are the height and width of the image, and T is the number of temporal observations (e.g. monthly composites or daily acquisitions).

To prepare these data for use in supervised or unsupervised machine learning, each pixel time series $X_{i,j} = (x_{i,j,1}, x_{i,j,2}, \dots, x_{i,j,T})$ is transformed into a feature vector:

$$\vec{x}_{i,j} = \big(f_1(X_{i,j}), f_2(X_{i,j}), \dots, f_M(X_{i,j})\big)$$

where each f_m is a time series feature extraction function (e.g. mean, variance, trend, autocorrelation), and M is the total number of extracted features.

A visual representation of this transformation is shown in Figure 1.

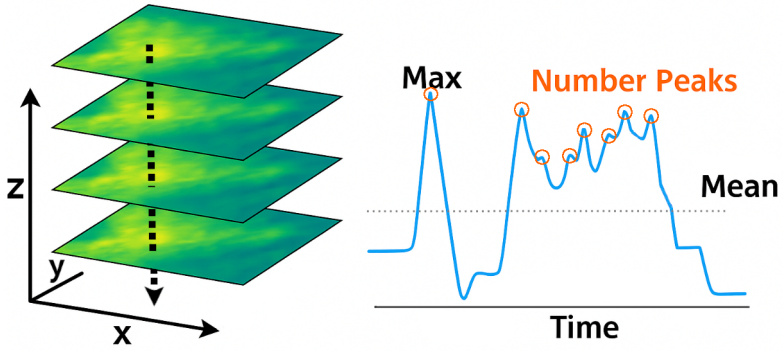

Figure 1: Feature Extraction Process

This results in a 2D design matrix of features for the entire image:

$$\mathbf{X}_{\text{features}} \in \mathbb{R}^{H \times W \times M}$$

This transformation effectively reduces the temporal complexity while preserving informative temporal patterns, enabling efficient training of models or aggregation to coarser units (e.g., fields or regions).

Additional static features (e.g., soil type, elevation), can be concatenated:

$$\vec{x}_{i,j}^{\text{final}} = \left[\vec{x}_{i,j} \,|\, \vec{a}_{i,j}\right] \in \mathbb{R}^{M+U}$$

where $\vec{a}_{i,j} \in \mathbb{R}^U$ represents the U univariate attributes at pixel (i, j).

Time Series Feature Set

The documentation summarizes the suite of time series features extracted by the `xr_fresh` module from gridded data. These features are designed to characterize the temporal behavior of each pixel (x_i, y_j). By including a diverse set of statistical, trend and distribution-based metrics, `xr_fresh` enables a detailed and scalable analysis of temporal patterns (Jin et al. 2022; Venkatachalam et al. 2024). Additional features can be added to the library as needed, and users can also define custom feature extraction functions.

Interpolation

The `xr_fresh` library includes functionality to interpolate missing values pixelwise in gridded data. The interpolation methods implemented in `xr_fresh` are designed to be computationally efficient and can handle large datasets effectively. The module supports advanced interpolation techniques including linear, nearest-neighbor, cubic, and univariate spline interpolation (Virtanen et al. 2020).

Formally, for a fixed pixel (i, j), let the time series be:

$$X_{i,j} = (x_{i,j,1}, x_{i,j,2}, \ldots, x_{i,j,T})$$

where some $x_{i,j,t}$ may be missing due to clouds or sensor gaps. The interpolation estimates these missing values by fitting a function $f(t)$ to the observed time steps $\{t_k \in [1, T] \mid x_{i,j,t_k} \text{ is observed}\}$. The interpolated value at time t is:

$$\hat{x}_{i,j,t} = f(t), \quad \text{for } x_{i,j,t} \text{ missing}$$

The function $f(t)$ may take the form of: 1) linear interpolation, 2) nearest neighbor, 3) cubic spline interpolation, or 4) univariate spline interpolation. If acquisition times are irregular, the time t is replaced by a datetime indexes.

Dimensionality Reduction

For high-dimensional inputs or when the number of bands/time steps is large, dimensionality reduction can improve model performance. `xr_fresh` integrates a GPU/CPU-parallelized Kernel Principal Component Analysis (KPCA) module (Pedregosa et al. 2011). The KPCA implementation samples valid observations for training, fits the kernel model, and projects each pixel's time series into a lower-dimensional space.

Software Framework

`xr_fresh` achieves scalability by employing a combination of parallel and distributed computing strategies. During feature extraction, functions are applied in parallel across spatial windows with Dask and xarray, which provide lazy evaluation, chunked computation (Graesser and Mann 2025; Hoyer and Hamman 2017; Rocklin 2015). Seamlessly integrated into the parallel pipeline and can leverage accelerated libraries like JAX, NumPy, ray, numba or PyTorch for additional speedup (Bradbury et al. 2018; Lam, Pitrou, and Seibert 2015; Harris et al. 2020; Moritz et al. 2018). Together, these strategies ensure that methods are highly scalable, for use on large-scale datasets.

Example: Precipitation In Africa

We apply `xr_fresh` methods to a dataset of monthly precipitation estimates in Africa (Figure 2) (Funk et al. 2015). The goal is to extract features from the time series data, enabling subsequent analysis and modeling. The `extract_features_series` function takes a list of files, a dictionary of desired features.

```
# create list of desired series and arguments
feature_list = {
    "minimum": [{}],
    "abs_energy": [{}],
    "doy_of_maximum": [{"dates": dates}],
    "mean_abs_change": [{}],
    "ratio_beyond_r_sigma": [{"r": 1}, {"r": 2}],
    "symmetry_looking": [{}],
    "sum": [{}],
    "quantile": [{"q": 0.05}, {"q": 0.95}],
}
from xr_fresh.extractors_series import extract_features_series

# Extract features from the geospatial time series
extract_features_series(image_list, feature_list, band_name, out_dir,
                        num_workers=12, nodata=-9999)
```

The extracted features found in Figure 3 can then be used in a variety of appli-

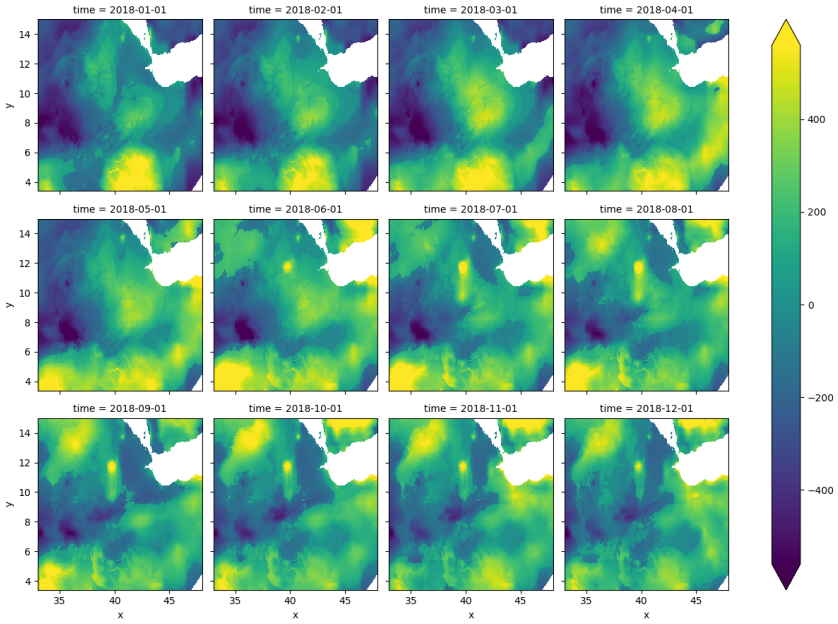

Figure 2: Precipitation input data

cations.

Conclusions

`xr_fresh` is a powerful and efficient tool for automated feature extraction from gridded time series. Using advanced statistical methods and parallel computing, it enables the extraction of a comprehensive set of features that can significantly enhance the performance of machine learning models. Integration with existing Python geospatial libraries ensures that `xr_fresh` is easy to use and can be seamlessly incorporated into existing machine learning workflows. It also provides advanced interpolation and dimensionality reduction capabilities, addressing common challenges in remote sensing data analysis.

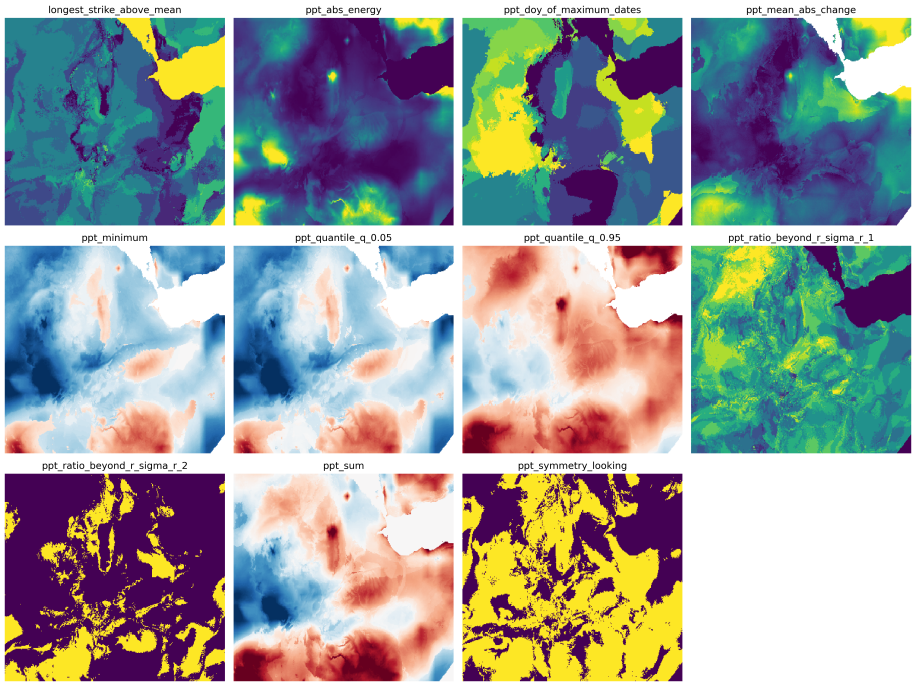

Figure 3: Time series feature set

References

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. 2018. *JAX: Composable Transformations of Python+NumPy Programs* (version 0.3.13). http://github.com/jax-ml/jax.

Christ, Maximilian, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. "Time Series FeatuRe Extraction on Basis of Scalable Hypothesis Tests (Tsfresh – a Python Package)." *Neurocomputing* 307: 72–77. https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067.

Delince, J, G Lemoine, P Defourny, J Gallego, A Davidson, S Ray, O Rojas, J Latham, and F Achard. 2017. "Handbook on Remote Sensing for Agricultural Statistics." *GSARS: Rome, Italy*. https://doi.org/10.13140/RG.2.2.13259.69920.

Faouzi, Johann. 2022. "Time Series Classification: A Review of Algorithms and Implementations." *Machine Learning (Emerging Trends and Applications)*. https://doi.org/10.5772/intechopen.1004810.

Funk, Chris, Pete Peterson, Martin Landsfeld, Diego Pedreros, James Verdin, Shraddhanand Shukla, Gregory Husak, et al. 2015. "The Climate Hazards Infrared Precipitation with Stations—a New Environmental Record for Monitoring Extremes." *Scientific Data* 2 (1): 1–21. https://doi.org/10.1038/sdata.2015.66.

Graesser, Jordan, and Michael Mann. 2025. *GeoWombat (V2.1.22): Utilities for Geospatial Data*. Zenodo. https://doi.org/10.5281/zenodo.15483823.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585: 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hoyer, S., and J. Hamman. 2017. "Xarray: N-D Labeled Arrays and Datasets in Python." *J. Open Res. Software*. https://doi.org/10.5334/jors.148.

Hufkens, Koen, Eli K Melaas, Michael L Mann, Timothy Foster, Francisco Ceballos, Miguel Robles, and Berber Kramer. 2019. "Monitoring Crop Phenology Using a Smartphone Based Near-Surface Remote Sensing Approach." *Agricultural and Forest Meteorology* 265: 327–37. https://doi.org/10.1016/j.agrformet.2018.11.002.

Jin, Guangyin, Fuxian Li, Jinlei Zhang, Mudan Wang, and Jincai Huang. 2022. "Automated Dilated Spatio-Temporal Synchronous Graph Modeling for Traffic Prediction." *IEEE Transactions on Intelligent Transportation Systems* 24 (8): 8820–30. https://doi.org/10.1109/TITS.2022.3195232.

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert. 2015. "Numba: A LLVM-Based Python JIT Compiler." In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2833157.2833162.

Li, Xixi, Yanfei Kang, and Feng Li. 2020. "Forecasting with Time Series Imaging." *Expert Systems with Applications* 160: 113680. https://doi.org/10.101

6/j.eswa.2020.113680.

Mann, Michael L., and James M. Warner. 2017. "Ethiopian Wheat Yield and Yield Gap Estimation: A Spatially Explicit Small Area Integrated Data Approach." *Field Crops Research* 201: 60–74. https://doi.org/https://doi.org/10.1016/j.fcr.2016.10.014.

Mann, Michael L, James M Warner, and Arun S Malik. 2019. "Predicting High-Magnitude, Low-Frequency Crop Losses Using Machine Learning: An Application to Cereal Crops in Ethiopia." *Climatic Change* 154 (1): 211–27. https://doi.org/10.1007/s10584-01902432-7.

Moritz, Philipp, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, et al. 2018. "Ray: A Distributed Framework for Emerging AI Applications." In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561–77. Carlsbad, CA: USENIX Association. https://doi.org/10.48550/arXiv.1712.05889.

Mumuni, Alhassan, and Fuseini Mumuni. 2024. "Automated Data Processing and Feature Engineering for Deep Learning and Big Data Applications: A Survey." *Journal of Information and Intelligence*. https://doi.org/10.1016/j.jiixd.2024.01.002.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30. https://doi.org/10.5555/1953048.2078195.

Rocklin, Matthew. 2015. "Dask: Parallel Computation with Blocked Algorithms and Task Scheduling." In *Proceedings of the 14th Python in Science Conference*, 130–36. https://doi.org/10.25080/Majora-7b98e3ed-013.

Venkatachalam, Sairam, Disha Kacha, Devarsh Sheth, Michael Mann, and Amir Jafari. 2024. "Temporal Patterns and Pixel Precision: Satellite-Based Crop Classification Using Deep Learning and Machine Learning." George Washington University, Department of Geography & Environment; Data Science Program.

Virtanen, Pauli, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17 (3): 261–72. https://doi.org/10.1038/s41592-019-0686-2.