# ADVANCING REPRODUCIBILITY AND INTEROPERABILITY IN CLOUD-BASED ANALYSIS OF EARTH OBSERVATION DATA CUBES

BRIAN PONDI

Doctoral Thesis Proposal

Institute for Geoinformatics
Faculty of Mathematics and Natural Sciences
University of Münster

May 2024

## CONTENTS

# 1 INTRODUCTION

Advancements in Satellite technology have enabled our global understanding of Earth by collecting extensive data on its surface, atmosphere, and oceans. Recently, more open governmental data-sharing policies have facilitated wider access to these large datasets [25], so that both the public and the scientific community can utilize them for environmental monitoring and to deepen our understanding of global phenomena such as food security, health, carbon biomass assessment, land cover mapping, and disaster management [7]. Consequently, this data is increasingly being hosted on cloud computing environments which has led to the development of several Earth Observation (EO) cloud computing platforms [4] such as Google Earth Engine (GEE) [6] and Sentinel Hub[1], enabling users to access, process and analyze large scale Earth observation data [4].

However, transitioning between these EO cloud platforms remains a challenge as they have different API interfaces, and different underlying complex software architectures, some are proprietary software and they have different EO data collections. This hinders reproducibility and interoperability in cloud-based EO data analysis, which are key concepts that are being adopted in the scientific community to advance research and collaboration. The concept of "research reproducibility" involves obtaining the same results as those reported in a scientific paper by using identical source code and data [5]. Recommendations to address technical issues that hinder reproducibility include embedding library versions in the source code, writing encapsulating code, running code in clean environments like the Docker container, and sharing code and data in their original forms [8]. Additionally, creating reproducible workflows is suggested to achieve reproducibility in Geography and Geosciences [14].

The concept of "interoperability" is defined by Wegner [24] as "the ability of two or more software components to cooperate despite differences in language, interface, and execution platform". Wagemann et al. [23] conducted a survey of 231 users of big Earth data, finding that about 90 % of the respondents identified "interoperability of data and data systems" as the most crucial aspect of data analysis.

The openEO specification [20], addresses the challenges of reproducibility and interoperability in EO cloud-based backends by standardizing how backend services are accessed irrespective of the underlying software architecture. It utilizes the Representational State Transfer (REST) protocol alongside Python, R, and Javascript clients

1 https://www.sentinel-hub.com/

— the top three programming languages favored by Big Earth data users, with Python and R being the most dominant [23]. The openEO ecosystem uses SpatioTemporal Asset Catalogs (STAC)[2] Specification for search and discovery of EO data, and data cubes as multidimensional arrays for representing satellite image time series.

Although openEO has advanced reproducibility and interoperability, it currently lacks a stable ML specification. Data cubes can be viewed as tensor representations, simplifying the application of ML algorithms such as Convolutional Neural Networks (CNNs) and related architectures [12]. Recent years have seen the emergence of ML algorithms like TempCNN [16], ConvLSTM [19], Temporo-Spatial Vision Transformer (TSViT) [22], spatial-spectral-temporal neural network (SSTNN) [17] and Temporal Self-Attention [3] that can be used to derive value from EO data cubes. Despite being open-source, configuring openEO backends for personal use remains complex due to their complex software architecture, exemplified by the scala-based[3] backend used by VITO OpenEO[4] that uses Spark-based Geotrellis engine[5]. Moreover, there is a need for easy search and discovery of ML models for EO data cubes [18], particularly models that are beneficial to communities with limited resources such as computing access and internet connectivity [13].

This study aims to expand the openEO ecosystem by developing ML API Specification for EO data cubes. The primary objectives are to have a specification covering data pre-processing, managing training datasets, training ML models, making predictions, uncertainty quantification, and mechanisms for saving and loading models. It shall support both traditional machine learning and advanced deep learning techniques. Additionally, the study aims to develop openEO backend systems with simplified architecture, facilitating easier deployment by EO data users and the scientific community in cloud environments where the data resides. Finally, the study shall also establish a blueprint for FAIR (Findable, Accessible, Interoperable, Reusable) ML models in Earth Observation.

2 https://stacspec.org/en
3 https://www.scala-lang.org/
4 https://github.com/Open-EO/openeo-geotrellis-extensions
5 https://geotrellis.io/

# 2 BACKGROUND

## 2.1 CHALLENGES AND CONSIDERATIONS IN STANDARDIZING ML APIS

The development and implementation of standardized ML API and spatio-temporal models catalog in the field of EO encompass complex challenges and considerations that must be effectively managed to ensure success. Overcoming these barriers requires technical knowledge, strategic planning, and collaborative efforts.

A primary challenge is the need for a standardized approach for sharing ML training data for EO data cubes, including support for filtering, sampling, and selecting training data. Another critical aspect is the interoperability between popular ML frameworks like Torch[1] and TensorFlow[2], along with their various versions, across programming languages such as Python and R. This will allow the EO community to share models and integrate them into different EO backend services. A common format, such as the Open Neural Network Exchange (ONNX)[3], is needed to facilitate this interoperability.

The new state-of-the-art DL methods developed for EO data cubes are complex and require extensive programming and DL expertise. These advanced approaches should be encapsulated in a simple API to democratize adoption and reuse within the EO community. Additionally, classical ML APIs often have different naming conventions for parameters across various libraries and their respective programming languages, which must be aligned for consistency and ease of use.

Model calibration presents another consideration, with a decision required on whether it should be conducted locally or in the cloud. This depends on factors, such as data accessibility, computational resources, security considerations, and collaboration requirements.

Classical ML methods, such as Random Forests, Support Vector Machines, and XGBoost, typically work with tabular data for inference and prediction. Therefore, it is crucial to outline the process of transforming EO data cubes into tabular formats. Handling training data spatial geometries, such as averaging values within polygons, is also essential for accurate model training and predictions.

Additionally, it is beneficial to consider and learn from widely used ML libraries across various programming languages. Notable exam-

1 https://pytorch.org
2 https://www.tensorflow.org
3 https://onnx.ai

ples include mlr3 [11], caret [9], and tidymodels [10] in R, scikit-learn [2, 15] in Python, MLJ [1] in Julia, and Keras[4] in both R and Python.

This study will focus on standardizing ML APIs for EO data cubes, offering reference implementations on two openEO-compliant backend services and developing easy-to-install, extendable openEO-compliant software. These efforts aim to make ML and DL techniques for EO data cubes more accessible, promoting collaboration, streamlining workflows, and improving the efficiency of EO data analysis.

## 2.2 SPATIO-TEMPORAL ML MODEL CATALOGING

Developing a spatio-temporal ML models catalog is a crucial component of the standardized ML API framework designed for EO data cubes. This catalog will serve as a centralized repository to organize, document, and provide access to diverse ML models tailored for EO data cubes analysis. This initiative will make it simpler for researchers and EO professionals to discover and apply appropriate models to different EO tasks, thereby speeding up the development process and promoting the sharing of effective methodologies within the community.

The catalog will provide detailed metadata for each model, including its type, input, output, hyperparameters, distinct name, architecture, and intended applications. For example, a crop type classification model should not be used for tree species classification, a land cover model for Eastern Africa is unsuitable for Western Europe, and a model created using Sentinel 2 is not ideal for Landsat 8. The catalog will also incorporate version control to manage updates efficiently. Fully integrated with the ML API, it allows for seamless model loading into user workflows and supports the contribution of new models and updates.

The EO community must adopt a comprehensive standard for cataloging ML models tailored to EO data. This standard should address the unique spatio-temporal characteristics of EO data cubes, ensuring that models are effectively categorized, accessible, and adaptable to both classical machine learning models and state-of-the-art deep learning models. Currently, there are two STAC extensions available: the STAC ML-Model Extension[5] and the more recent MLM STAC Extension[6]. The latter supports cataloging both classical ML and advanced DL techniques.

STAC has client libraries in popular programming languages within the EO community, such as Rstac [21] in R and Pystac[7] in Python.

4 https://keras.io/
5 https://github.com/stac-extensions/ml-model
6 https://github.com/crim-ca/mlm-extension
7 https://github.com/stac-utils/pystac

These libraries can be extended to allow users to query, access, and integrate ML models into their scripting workflows.

In this study, we will build a spatio-temporal ML models cataloging server dedicated to organizing and documenting a wide array of ML models suitable for EO data cubes through metadata. Our investigation will evaluate its practicality and examine the suitability of ONNX as a common ML format to ensure the reproducibility and interoperability of spatio-temporal ML models for EO data cubes across diverse ML libraries/frameworks and programming languages. Such efforts will contribute to the adoption of ML models cataloging within the EO community, facilitating the easier discovery and application of relevant models for diverse EO tasks. Ultimately, this initiative seeks to foster collaboration, encourage model reuse, and streamline workflows within the EO community.

# 3 RESEARCH QUESTIONS

The proposed dissertation aims to advance reproducibility and interoperability in cloud-based analysis of Earth observation data cubes. The following research questions are defined:

RQ 1 What core functionalities should be included in a standardized ML API for EO data cubes?

- RQ 1.1 How can the API specification facilitate advanced ML techniques like deep learning?
- RQ 1.2 How will spatial-temporal training data be handled and facilitated?

RQ 2 How can we perform computations, including those involving ML and DL algorithms, on large-scale EO data cubes in a cloud environment in an open and reproducible manner, while also ensuring the capability of local execution?

- RQ 2.1 What open-source tools and environments can be used to ensure that computations, including those involving ML and DL algorithms, on EO data cubes in the cloud can be reproduced and verified by others, and also executed locally?

RQ 3 How can the FAIR principles be integrated into ML models for EO data cubes?

- RQ 3.1 How does the STAC-ML[1] / MLM-STAC[2] extension(s) facilitate the findability and accessibility of EO data cubes ML models, and what are the challenges associated with its integration?
- RQ 3.2 How can we ensure consistent performance and interoperability of ML models across different frameworks like TensorFlow and PyTorch, including their varying versions?
- RQ 3.3 How can we achieve EO data cubes ML models interoperability across programming languages (i.e. R and Python)?

1 https://github.com/stac-extensions/ml-model
2 https://github.com/crim-ca/mlm-extension

# 4 WORK PACKAGES

## 4.1 WP1: ML API SPECIFICATION FOR EO DATA CUBES

We are developing a standardized Machine Learning (ML) Application Programming Interface (API) designed to extract value from Earth Observation (EO) data cubes using classical ML algorithms, including Random Forests, Support Vector Machines, XGBoost, and Deep Learning-based CNN models and attention models. The importance of this work lies in its potential to streamline and standardize the process of applying ML to EO data, which is often hampered by fragmented tools and workflows. By providing a unified API, we address the interoperability challenge, enabling transitions between different backend services and reducing the learning curve for users.

## 4.2 WP2: OPEN-SOURCE SOFTWARE TO ANALYZE EO DATA CUBES

This work package focuses on creating lightweight, open-source software that integrates STAC, Data Cubes, and the openEO standardized API. The design of this software targets easy deployment in cloud environments, which host large EO datasets, as well as local execution (i.e., independent of a cloud) through Docker images. The software will also include Machine Learning (ML) and Deep Learning (DL) algorithms to provide robust data analysis capabilities. By promoting open science and reproducibility, these software solutions will empower a broader community of researchers and data users to extract value from EO data and share their reproducible workflows.

## 4.3 WP3: FAIR ML MODELS FOR EO DATA CUBES

This work package focuses on handling, describing, and sharing trained models by applying the FAIR (Findable, Accessible, Interoperable, and Reusable) principles to ML models designed for EO data cubes. Specifically, we aim to incorporate either the STAC-ML[1] or MLM-STAC[2] extension(s) for model cataloging and utilize the ONNX framework to evaluate model interoperability across various ML frameworks (such as TensorFlow and PyTorch) and programming languages (R and Python). This approach ensures that models adhere to FAIR standards, making them more adaptable and usable within the scientific community.

1 https://github.com/stac-extensions/ml-model
2 https://github.com/crim-ca/mlm-extension

# 5 EXPECTED RESULTS AND EVALUATIONS

## 5.1 EXPECTED RESULTS

The following are the expected results of the dissertation:

- **Machine Learning API Specification in the OpenEO Ecosystem:** The dissertation is expected to deliver a specification for a Machine Learning API that integrates with the OpenEO ecosystem. This specification will detail the core functionalities needed to utilize various ML algorithms and ensure adaptability. The aim is to establish a standardized interface that fosters easier access, manipulation, and analysis of EO data cubes, facilitating broader adoption and interoperability within the scientific community.

- **Open-Source Software to Analyze EO Data Cubes:** An implementation of lightweight, RESTful software that aligns with the openEO API. It will serve as a reference implementation for openEO ML API specification, demonstrating the practical application of the API specification and providing a benchmark for future development. The software will be designed to be open-source, encouraging community contributions and the sharing of ideas to continually enhance its functionality and ease of use.

- **Blueprint for FAIR Machine Learning Models for EO Data Cubes:** Developing a blueprint that applies FAIR principles to Machine Learning models specific to EO data cubes is another expected outcome. This blueprint will outline approaches for ensuring these models are findable, accessible, interoperable, and reusable. Emphasis will be placed on cataloging models using the STAC-ML extension, ensuring interoperability through the ONNX framework, and documenting challenges across various backend services and programming languages.

## 5.2 EVALUATION

The following details the structured evaluation for each expected outcome:

### 5.2.1 *Evaluation of Machine Learning API Specification in the OpenEO Ecosystem*

- **Functionality Testing:** Conduct tests to verify that the listed functionalities of the ML API are implemented correctly and are operational across different backend services and EO cloud platforms.
- **Usability and Adaptability Tests:** Gather feedback from a diverse group of end-users, including data scientists and developers, to assess the API's ease of use and adaptability in real-world scenarios.

### 5.2.2 *Evaluation of Open-Source Software to Analyze EO Data Cubes*

- **Integration Tests:** Ensure integration with different data sources and the existing openEO ecosystem, along with compatibility across various operating systems and dependencies.
- **Community Feedback:** Monitor adoption and contributions from the community, including code contributions, issue tracking, and engagement in forums and discussions.

### 5.2.3 *Evaluation of blueprint for FAIR Machine Learning Models for EO Data Cubes*

- **Compliance with FAIR Principles:** Detailed metrics on findability, accessibility, interoperability, and reusability to assess conformity with FAIR guidelines.
- **Cross-framework and Cross-language Interoperability:** Testing the interoperability of EO data cubes ML models across different machine learning frameworks (TensorFlow, PyTorch) and programming environments (R, Python) using the ONNX framework.

# 6 EXPECTED CONTRIBUTIONS

## 6.1 PAPER CONTRIBUTIONS

During my research, I anticipate publishing at least four open-access papers:

1. openEO-ML: Machine Learning API Specification for Earth Observation Data Cubes Analysis.
2. FAIR ML Models for Earth Observation Data Cubes.
3. openEOcraft: A versatile openEO compliant backend.
4. * openEOcubes: An Open-Source and Lightweight R-Based RESTful Web Service for Analyzing Earth Observation Data Cubes.

   Available at https://doi.org/10.1007/s12145-024-01249-y.

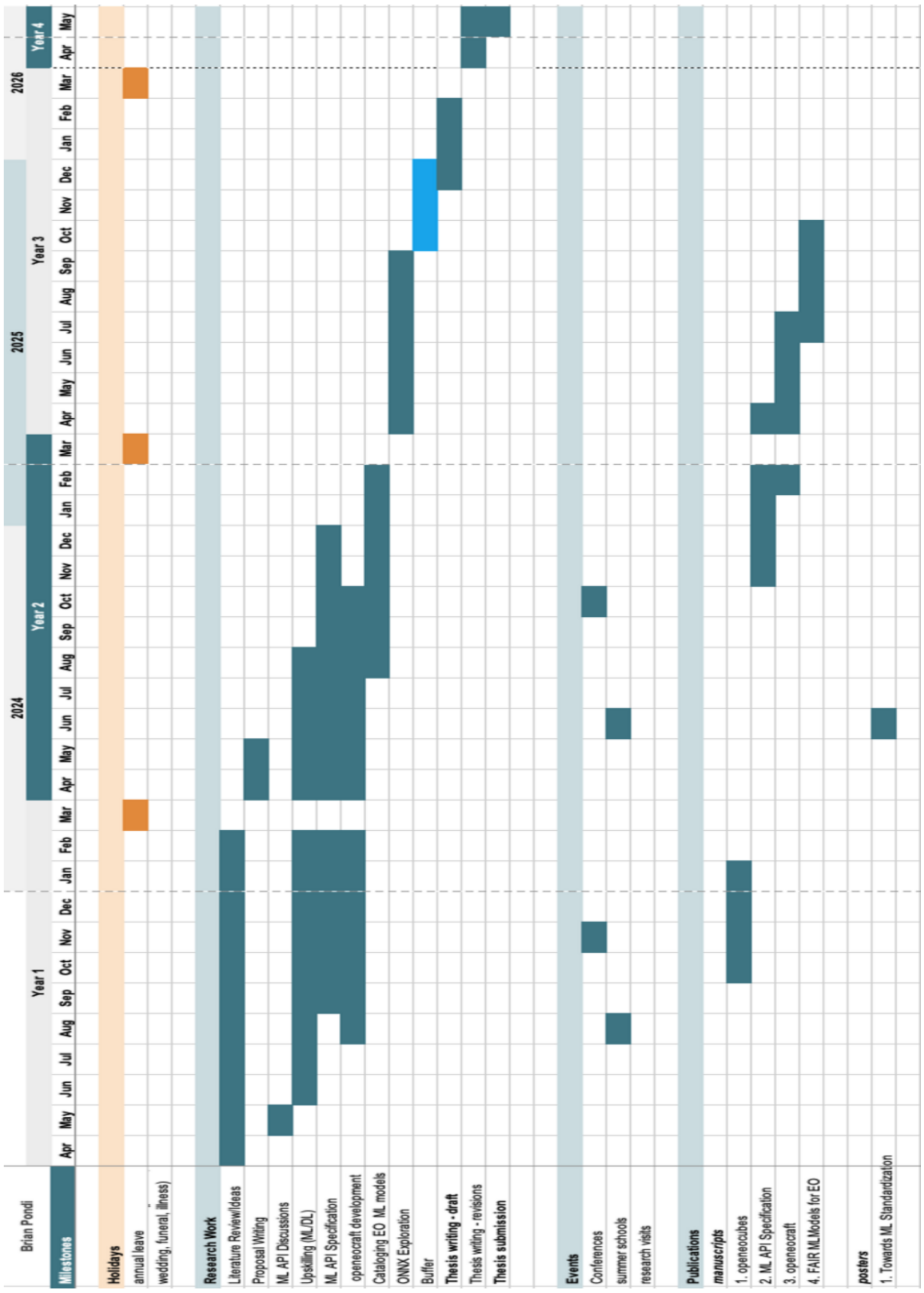
NB: * The last paper has already been published, originating from work started during my MSc and further developed and maintained during my Doctoral time. Additionally, ML API will be incorporated.

## 6.2 SOFTWARE CONTRIBUTIONS

During my research, I will develop and contribute to open-source software projects.

| **Software** | **Description** |
|---|---|
| openEO API | Incorporation of ML API Specification for Earth Observation (EO) data cubes within the openEO ecosystem. Accessible at https://github.com/Open-EO/openeo-api. |
| openeocubes | A lightweight, easy-to-deploy RESTful R backend designed for cloud environments. Available at https://github.com/PondiB/openeocubes. |
| openeocraft | A versatile openEO backend that supports various R packages, featuring reference ML API implementations. See https://github.com/Open-Earth-Monitor/openeocraft. |
| openearth-ml-server | A server dedicated to cataloging SpatioTemporal ML models. Found at https://github.com/PondiB/openearth-ml-server. |

# 7 TIMELINE



(a) Doctoral journey Gantt chart

# BIBLIOGRAPHY

[1] Anthony D Blaom, Franz Kiraly, Thibaut Lienart, Yiannis Simillides, Diego Arenas, and Sebastian J Vollmer. "MLJ: A Julia package for composable machine learning." In: *arXiv preprint arXiv:2007.12285* (2020).

[2] Lars Buitinck et al. *API design for machine learning software: experiences from the scikit-learn project*. 2013. arXiv: 1309.0238 [cs.LG].

[3] Vivien Sainte Fare Garnot and Loïc Landrieu. "Lightweight Temporal Self-Attention for Classifying Satellite Image Time Series." In: *CoRR* abs/2007.00586 (2020). arXiv: 2007.00586. URL: https://arxiv.org/abs/2007.00586.

[4] Vitor Conrado Faria Gomes, Gilberto Ribeiro de Queiroz, and Karine Reis Ferreira. "An Overview of Platforms for Big Earth Observation Data Management and Analysis." In: *Remote. Sens.* 12 (2020), p. 1253. URL: https://api.semanticscholar.org/CorpusID:216648139.

[5] Steven N Goodman, Daniele Fanelli, and John PA Ioannidis. "What does research reproducibility mean?" In: *Science translational medicine* 8.341 (2016), 341ps12–341ps12.

[6] Noel Gorelick, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. "Google Earth Engine: Planetary-scale geospatial analysis for everyone." In: *Remote Sensing of Environment* 202 (2017). Big Remotely Sensed Data: tools, applications and experiences, pp. 18–27. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2017.06.031. URL: https://www.sciencedirect.com/science/article/pii/S0034425717302900.

[7] Pratistha Kansakar and Faisal Hossain. "A review of applications of satellite earth observation data for global societal benefit and stewardship of planet earth." In: *Space Policy* 36 (2016), pp. 46–54. ISSN: 0265-9646. DOI: https://doi.org/10.1016/j.spacepol.2016.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0265964616300133.

[8] Markus Konkol, Christian Kray, and Max Pfeiffer. "Computational reproducibility in geoscientific papers: Insights from a series of studies with geoscientists and a reproduction study." In: *International Journal of Geographical Information Science* 33.2 (2019), pp. 408–429. DOI: 10.1080/13658816.2018.1508687.

[9] Max Kuhn. "Building predictive models in R using the caret package." In: *Journal of statistical software* 28 (2008), pp. 1–26.

[10] Max Kuhn and Hadley Wickham. “Tidymodels: Easily install and load the’tidymodels’ packages.” In: *R package version 0.1* 2 (2020).

[11] Michel Lang, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. “mlr3: A modern object-oriented machine learning framework in R.” In: *Journal of Open Source Software* 4.44 (2019), p. 1903.

[12] David Montero Loaiza et al. “Data Cubes for Earth System Research: Challenges Ahead.” In: *EarthArXiv* (2023). DOI: 10.31223/X58M2V. URL: https://doi.org/10.31223/X58M2V.

[13] Catherine Nakalembe and Hannah Kerner. “Considerations for AI-EO for agriculture in Sub-Saharan Africa.” In: *Environmental Research Letters* 18.4 (2023), p. 041002. DOI: 10.1088/1748-9326/acc476. URL: https://dx.doi.org/10.1088/1748-9326/acc476.

[14] Daniel Nüst and Edzer Pebesma. “Practical Reproducibility in Geography and Geosciences.” In: *Annals of the American Association of Geographers* 111.5 (2021), pp. 1300–1310. DOI: 10.1080/24694452.2020.1806028.

[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. “Scikit-learn: Machine learning in Python.” In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[16] Charlotte Pelletier, Geoffrey I Webb, and François Petitjean. “Temporal convolutional neural network for the classification of satellite image time series.” In: *Remote Sensing* 11.5 (2019), p. 523.

[17] Mengjia Qiao, Xiaohui He, Xijie Cheng, Panle Li, Haotian Luo, Lehan Zhang, and Zhihui Tian. “Crop yield prediction from multi-spectral, multi-temporal remotely sensed imagery using recurrent 3D convolutional neural networks.” In: *International Journal of Applied Earth Observation and Geoinformation* 102 (2021), p. 102436. ISSN: 1569-8432. DOI: https://doi.org/10.1016/j.jag.2021.102436. URL: https://www.sciencedirect.com/science/article/pii/S0303243421001434.

[18] Esther Rolf, Konstantin Klemmer, Caleb Robinson, and Hannah Kerner. *Mission Critical – Satellite Data is a Distinct Modality in Machine Learning*. 2024. arXiv: 2402.01444 [cs.LG].

[19] Marc Rußwurm and Marco Körner. “Convolutional LSTMs for Cloud-Robust Segmentation of Remote Sensing Imagery.” In: *CoRR* abs/1811.02471 (2018). arXiv: 1811.02471. URL: http://arxiv.org/abs/1811.02471.

[20] Matthias Schramm et al. “The openEO API-Harmonising the Use of Earth Observation Cloud Services Using Virtual Data Cube Functionalities.” In: *Remote. Sens.* 13 (2021), p. 1125. URL: https://api.semanticscholar.org/CorpusID:233195187.

[21] Rolf Simoes, Felipe Carvalho de Souza, Matheus Zaglia, Gilberto Ribeiro de Queiroz, Rafael D. C. dos Santos, and Karine Reis Ferreira. “Rstac: An R Package to Access Spatiotemporal Asset Catalog Satellite Imagery.” In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. 2021, pp. 7674–7677. DOI: 10.1109/IGARSS47720.2021.9553518.

[22] Michail Tarasiou, Erik Chavez, and Stefanos Zafeiriou. *ViTs for SITS: Vision Transformers for Satellite Image Time Series*. 2023. arXiv: 2301.04944 [cs.CV].

[23] Julia Wagemann, Stephan Siemen, Bernhard Seeger, and Jörg Bendix. “Users of open Big Earth data – An analysis of the current state.” In: *Computers & Geosciences* 157 (Aug. 2021), p. 104916. DOI: 10.1016/j.cageo.2021.104916.

[24] Peter Wegner. “Interoperability.” In: *ACM Computing Surveys (CSUR)* 28.1 (1996), pp. 285–287.

[25] Michael A. Wulder, Jeffrey G. Masek, Warren B. Cohen, Thomas R. Loveland, and Curtis E. Woodcock. “Opening the archive: How free data has enabled the science and monitoring promise of Landsat.” In: *Remote Sensing of Environment* 122 (2012). Landsat Legacy Special Issue, pp. 2–10. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2012.01.010. URL: https://www.sciencedirect.com/science/article/pii/S003442571200034X.