







A Scalable AI Training Platform for Remote Sensing Data

Hendrik M. Würz ^{1,2}, Kevin Kocon ^{1,2}, Barbara Pedretschner ³, Eva Klien¹, and Eva Eggeling ⁴

¹Fraunhofer Institute for Computer Graphics Research IGD, Fraunhoferstraße 5, Darmstadt, Germany

²Technical University of Darmstadt, Karolinenplatz 5, Darmstadt, Germany

³Fraunhofer Austria Research GmbH, KI4LIFE, Lakeside B13a, 9020 Klagenfurt, Austria

⁴Fraunhofer Austria Research GmbH, Visual Computing, Inffeldgasse 16c, 8010 Graz, Austria

Correspondence: Hendrik M. Würz (hendrik.martin.wuerz@igd.fraunhofer.de)

Abstract.

We present a platform to support the AI development lifecycle with focus on large data like remote sensing. We target developers who are not allowed to use existing commercial cloud platforms for legal reasons or data compliance. The flexible implementation of our platform enables a deployment on classic server infrastructures as well as on internal clouds. Our goals of scalable and resource-efficient execution, independence from specific AI frameworks and programming languages, as well as reproducibility of results are met through a workflow-based calculation combined with the tool Data Version Control. The capabilities of the platform are demonstrated by training an AI-based forest type classification.

Keywords. Artificial Intelligence, Workflow Management, On-Premises Calculation, Cloud computing, Remote Sensing

1 Introduction

Artificial intelligence systems have become more visible in remote sensing in the last years. For example, AI helps to predict yields (Jung et al. (2021)) or to detect deforestation (John and Zhang (2022)).

However, before an AI can be used, it must be trained. Especially in the field of remote sensing, this requires large amounts of storage. At the same time, ground truth data is rarely available and data policies demand for special protection. Sometimes, this excludes the use of commercial clouds. We present a platform that supports earth scientists in training AI models. Our platform is based on open software, can easily be installed on internal hardware, and still exploits the capabilities of (internal) clouds. The entire process from data preparation to the trained AI model is structured as a workflow. This ensures reproducibility

and enables resource-efficient computation. In this paper, we present our architecture, demonstrate an example as proof of concept and give an outlook on future work.

2 Related Work

There are commercial as well as open source products to support AI development. The most prominent training platforms include Google AI Platform (Google (2023a)) or its successor Vertex AI (Google (2023c)), the Azure AI Platform from Microsoft (Microsoft (2023)) and Amazon's SageMaker (Amazon Web Services (2023)). They are all tightly coupled with their cloud computing platforms. This enables massive scaling but might prevent the usage for applications with data policy issues. Further, to cover the complete pipeline from raw data to AI, additional systems are needed. The Google Earth Engine GEE (Google (2023b)) for example is a powerful platform for manipulating and pre-processing data. Additionally, analysis ready data can be ingested from external sources, which can significantly reduce the effort before training an AI. For Earth Observation (EO) applications, there are several EO data providers and platforms that also offer pre-processed datasets for training (e.g. (Planet Labs (2023b, c))). Having worked with the aforementioned tools, we see a clear benefit for an integrated solution for pre-processing and training as proposed in this paper. Besides the large cloud providers, smaller solutions for AI development can be found. Especially in medicine, systems with a focus on data security have been developed. Cohen and Kovacheva (2021) presented MERLIN, a platform that supports physicians in data analysis. Many of their ideas, such as the construction of a modular system, can also be applied to remote sensing. Kadri et al. (2022) for example emphasizes the importance of containerization in data processing - ideas we take up and combine with a scientific workflow management sys-

tem. Other projects like Kubeflow Pipelines follow a similar idea (Kubeflow Authors (2023)). However, they are closely linked to respective AI frameworks such as TensorFlow (Google (2023d)). This achieves a great integration, but it limits the flexibility with respect to other frameworks like PyTorch (Linux Foundation (2023)).

3 Architecture

In the field of remote sensing, data processing developers often have to handle large data and follow tight data policies. In addition, they also want to work productively. This leads to the following requirements:

R1) On-Premises. For data compliance reasons, calculations often must take place on-premises. The platform should be installable on internal infrastructure and be based on open technology.

R2) Flexibility. Different calculation steps require different hardware such as large disks or GPU acceleration. The platform should allocate appropriate resources in each step and release them afterwards to ensure a cost-efficient execution.

R3) Reproducibility and versioning. In academic context, many experiments are often repeated with different parameters. The platform should document, which configuration led to which result to support reproducibility.

R4) Language agnostic. Developers should be able to use their favorite programming language or framework. The platform should not make any restrictions.

Our architecture has to address all these requirements. For requirement (R2), scientific workflow management systems (WMS) are beneficial (Juve and Deelman (2010); Lin et al. (2009); Liu et al. (2016)). They model tasks as a workflow in which multiple services are invoked. Resource requirements can be defined for each service and the WMS provides them at run time. This way, the developer retains control over the service implementation. At the same time, the WMS can control the execution.

There are many open source WMS like Pegasus (Deelman et al. (2015)), Apache Airflow (Apache Software Foundation (2023)) or Argo (Argo Project Authors (2023)). For our platform, we use the open source WMS Steep because it scales well (Krämer (2021)), offers a powerful workflow definition language (Krämer et al. (2021)) and is easy to set up. Furthermore, it can be deployed in a cloud environment as well as on individual servers. This way, requirement (R1) is fulfilled. When Steep is running in a cloud, it can start and stop VMs on demand, depending on the service to be executed. This leads to a cost-efficient execution and provides flexibility regarding various hardware requirements.

For requirement (R4), we implement container-based processing and use Docker images for the services in our workflows. (R3) is ensured by documenting both the workflows and the data used (see Section 3.2).

All components of our platform are summarized in Fig. 1. There are five scenarios in which people interact with the platform: New services can be added to the platform, an AI model can be trained or used, a final workflow can be published, and customers can use AI models. In the following sections, we describe each of these situations and how the platform supports the process.

3.1 Service Deployment

New services can be added to the platform to extend its functionality. Examples include access to external APIs, format conversion, data processing or additional AI frameworks.

Any service is managed in a Git repository as shown in step ① in Fig. 1. We distinguish between three types of services: static data processing services (orange), services to train neural networks (blue) and services to use pre-trained networks (green). However, from a technical point of view, all services are treated identically. Each service is a repository containing a Dockerfile. When the repository changes, a new Docker image is automatically built by a Gitlab runner and pushed to the Docker registry ②. By using Docker images, the WMS starts a container without having to deal with different runtime environments. The service developer maintains dependencies inside the image and changes have no side effects on other services.

When the Docker images are available, the service is registered in the WMS ③. This is not limited to self-built images, public ones can be referenced as well. The registration specifies the service interfaces such as parameters, input and output files, as well as the resource requirements.

3.2 Model Training

Once all required services are deployed, they can be assembled into a workflow ④. To train an AI model, input data needs to be prepared first. The platform itself does not impose any requirements on the data. It can be raw files in an object storage, APIs to third party systems, databases, or anything else. Only the developed services have to comply with the data. Data preparation may include steps like download, filter or conversion, illustrated in orange in Fig. 1. After these static processing steps, the actual training follows (blue). For each AI framework such as PyTorch or TensorFlow, there is a corresponding training service. If other frameworks are needed, new services can be added as described in Section 3.1.

For training, the developer specifies the AI model structure within a separate file ⑤. It is not part of the training service to keep it reusable. Instead, it is referenced by a parameter and loaded at run time.

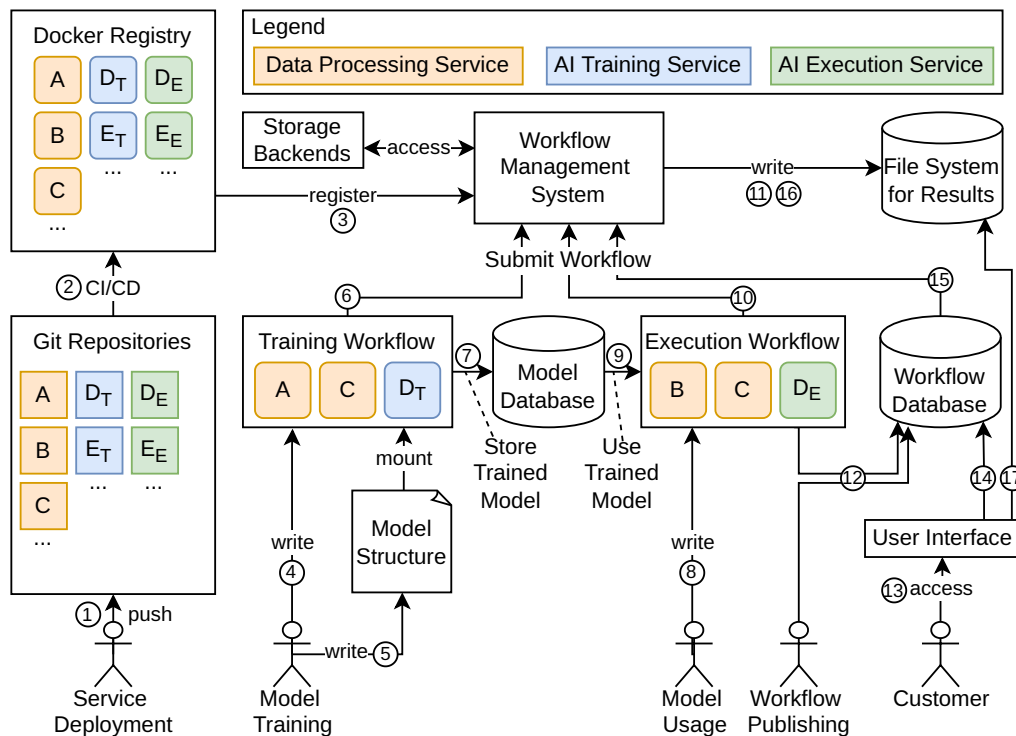


Figure 1. Architecture of the platform. There are five scenarios in which a user interacts with the platform. This figure illustrates what happens in each of them.

The complete workflow is sent to the WMS (6), which executes it on suitable hardware using the service specification from Section 3.1. The WMS also takes care of data exchange between individual services, parallelizes execution, cleans up temporary files, displays error logs and can initiate retries. The latter is particularly helpful when accessing external data sources that might be unavailable temporarily or in situations where access limits have been reached.

For the training, our platform ensures requirement (R3) “reproducibility and versioning” on two levels. First, the WMS documents the input and output paths of each service along with its version number. However, when performing AI training, the model structure file is just a parameter to the service and might be modified between executions. In this case, the second versioning mechanism applies: The tool Data Version Control (DVC) links data and model of experiments (iterative.ai (2023)). Even if the model structure changes, DVC tracks what a network was trained with. DVC can be used via a command line program, making it easy to integrate into the training workflow. It stores its data in a Git repository so that it can be retrieved from there later. Finally, the training results are pushed in a model database (7).

3.3 Model Usage

To use an AI model, some service invocations can be copied from the training workflow. Steps like downloads and format conversions of input data remain, but no

ground truth data need to be prepared. Additionally, the training service (blue) is replaced by an execution service (green) (8). Similar to training, this service only contains libraries to run a framework like TensorFlow. The actual AI model is referenced by a parameter (9) and loaded from the model database at run time.

The user submits the workflow to the WMS (10), which manages the execution as described above. The AI execution service performs a forward propagation of the prepared input data through the referenced model and writes the outputs on a separate file system (11).

3.4 Exploitation

When a workflow is ready for delivery, it is published in a workflow database (12). This defines the used services, their order and the version of AI models. However, the paths to input data and configurations are still parameterized so that they can be defined by the customer later.

To do this, the customer accesses the workflow database via a user interface (13), selects a workflow and specifies the missing values (14). The data on which the model should be executed can either be uploaded or made available via an API depending on what the services in the workflow expect. The final workflow is sent to the WMS (15). When the calculation is complete (16), the results can be downloaded (17).

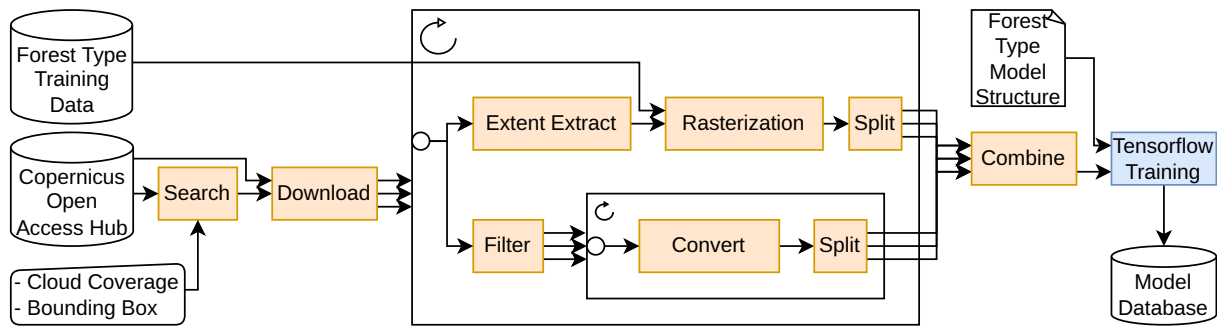


Figure 2. Training Workflow. Prepares ground truth and input data, and trains the AI model with TensorFlow.

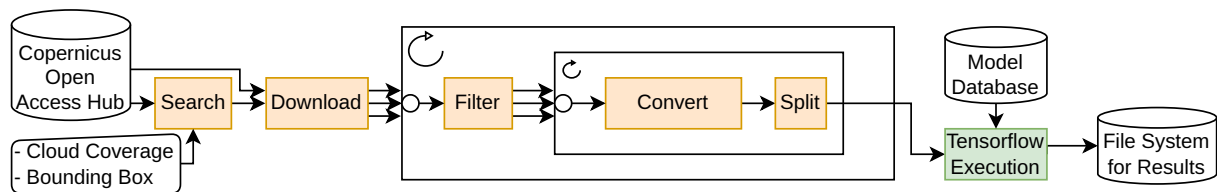


Figure 3. Execution Workflow. The steps to prepare the input data are nearly the same as in Fig. 2. Large parts can be reused. However, the AI model is now used instead of trained.

4 Evaluation

We used the platform to determine forest types based on Sentinel 2 data. For details on the used CNN models, the hyper-parameters and the final results, we refer to Kocon et al. (2022). In this section, we summarize the developed services and our experience with the platform. Fig. 2 shows the workflow to train the model. It should be noted that we used this workflow to train different CNN models with almost no overhead. For more details we refer again to Kocon et al. (2022). We started by searching for relevant Sentinel 2 images in the Copernicus Open Access Hub (ESA (2023)). Criteria were the cloud coverage and a bounding box of the relevant area. The information were passed as parameters to the workflow. This way, they can be changed easily to train on another region. The identified satellite images were downloaded and processed in parallel. Instead of Sentinel 2, other data sources could be accessed, such as Landsat data (Williams et al. (2006)), or the high-resolution images from Planet (Planet Labs (2023a)) by replacing the *download* service. For each image, its extent was extracted, corresponding ground truth data was fetched from an external API and rasterized.

The following *Split* service split the generated ground truth image into tiles, ensuring the resolution required by the AI model.

Additionally, every downloaded image was filtered to remove unneeded frequency bands, converted to PNG and cropped into tiles like the ground truth image. The service *Combine* copies the pairs of ground truth and satellite images into a common directory. The final TensorFlow service took this directory together with a model definition and learned how different forest types look in satellite images. During the development of our AI model, we exe-

cuted the pipeline several times with slight modifications. For example, we tested which frequency bands are important, or what influence seasons have on the classification. All of these tests were managed by the platform and we could directly compare the accuracy of the trained networks.

As explained in Section 3.2, we stored the trained model in a model database. Afterwards, we could use it with another workflow (see Fig. 3). As before, we downloaded Sentinel 2 images for a desired region, extracted the necessary frequency bands, and split the images into small tiles. However, instead of using the output tiles for training, they were now passed to an AI execution service. The final results with the forest type classification were written to the file system and are ready for download.

5 Discussion

In this section, we discuss some decisions in platform design. We address partial execution of workflows and options for access rights.

Mapping the entire pre-processing and training pipeline in one workflow sounds tempting. All steps from raw data, to a final trained AI model are formalized and reproducible. However, practice shows that often only parts of the workflow are executed. Service developers test their service in an isolated workflow, AI developers create necessary training data once and then just run the training service. Consequently, the *entire workflow* is started only in the exploitation phase by the customer. An extended platform design could take this into account. Here, developers would always send the entire workflow, but the platform checks if each service really needs to be executed or if data from a

previous execution can be reused. If this is the case, the corresponding services are skipped. Otherwise, a service or parameter has changed and the calculations are needed. However, during the design phase we explicitly decided not to do this, because we wanted to keep the platform simple. Currently, everything in the workflow is executed. The more intelligence we add, the more complicated it becomes. Nevertheless, there may be development teams that would like to have such a feature. In these cases, an appropriate plugin could be written for our workflow management system Steep.

No user or rights management is implemented within our platform. Instead, the network in which the platform is running has to be secured. Everybody who has access to the network is allowed to run any service and access all data. In situations where more restrictions were needed, another instance of the platform was deployed with only the allowed services included. This keeps the maintenance effort low as no user registrations, nor granting of rights are needed. Since the platform addresses individual companies and not large, public cloud providers, this approach can also be practical for others. However, there are also situations in which increased security requirements prevent such a solution.

6 Conclusion & Future Work

In this paper, we presented a platform to support the development of AI models based on large data like remote sensing. For this purpose, we used a workflow management system (WMS) with automated deployment pipelines for services and introduced versioning. Additionally, we demonstrated the usability of the platform for forest type classification.

The platform fulfills the requirements from Section 3. All components of the platform can be executed on premises since we are only using open source technology (R1). We model all computations as a workflow with multiple services. Each service has individual hardware requirements, ensuring that only the necessary resources are allocated. For this, the WMS Steep starts and stops appropriate VMs and monitors the execution of tasks (R2). The services are encapsulated in Docker images and thus are independent towards the used programming language (R4). The platform supports versioning of all calculations (R3) by documenting all inputs and outputs of a service, while changing data is versioned by DVC. In this way, calculations remain reproducible and can be analyzed later.

In the future, we plan to support AI training even more. The underlying WMS enables us to scale the resources easily. Especially for hyperparameter tuning (Feurer and Hutter (2019)) this can be an advantage. A workflow can test several parameters simultaneously and then further optimize the most promising ones.

Another interesting feature is user steering (Mattoso et al. (2013)). The term comes from the scientific workflow research and describes the intervention of humans in a running workflow. In combination with human-in-loop training (Mosqueira-Rey et al. (2022)), new exciting fields of research questions arise. How can versioning be ensured in this case? How does the user interact with a distributed running training? In the future we want to address these questions and extend the platform accordingly.

7 Data and Software Availability

Scripts to set up the platform initially are available at <https://github.com/igd-geo/ai-training-platform/releases/tag/v1.0.0>. The used workflow management system Steep is an open source project <https://steep-wms.github.io/>.

Acknowledgements. This work is part of the strategic project “Die Digitalisierung des Waldes” funded by Fraunhofer. Thanks to Alexander Steinhardt and Philipp Fleiß for the valuable discussions.

References

- Amazon Web Services: Machine Learning, <https://aws.amazon.com/sagemaker/>, last visited: 2023-02-10, 2023.
- Apache Software Foundation: Apache Airflow, <https://airflow.apache.org/>, last visited: 2023-01-18, 2023.
- Argo Project Authors: Argo Workflows, <https://argoproj.github.io/workflows/>, last visited: 2023-01-18, 2023.
- Cohen, R. Y. and Kovacheva, V. P.: A Methodology for a Scalable, Collaborative, and Resource-Efficient Platform to Facilitate Healthcare AI Research, CoRR, abs/2112.06883, <https://arxiv.org/abs/2112.06883>, 2021.
- Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., and Wenger, K.: Pegasus: a Workflow Management System for Science Automation, *Future Generation Computer Systems*, 46, 17–35, <https://doi.org/10.1016/j.future.2014.10.008>, 2015.
- ESA: Open Access Hub, <https://scihub.copernicus.eu/>, last visited: 2023-02-22, 2023.
- Feurer, M. and Hutter, F.: Hyperparameter optimization, Automated machine learning: Methods, systems, challenges, pp. 3–33, 2019.
- Google: Introduction to AI Platform | Google Cloud, <https://cloud.google.com/ai-platform/docs/technical-overview>, last visited: 2023-02-06, 2023a.
- Google: Google Earth Engine, <https://earthengine.google.com/>, 2023b.
- Google: Vertex AI | Google Cloud, <https://cloud.google.com/ai-platform/docs/technical-overview>, last visited: 2023-02-06, 2023c.

- Google: TensorFlow, <https://www.tensorflow.org/>, last visited: 2023-02-01, 2023d.
- iterative.ai: Data Version Control · DVC, <https://dvc.org/>, last visited: 2023-01-18, 2023.
- John, D. and Zhang, C.: An attention-based U-Net for detecting deforestation within satellite sensor imagery, *International Journal of Applied Earth Observation and Geoinformation*, 107, 102 685, <https://doi.org/10.1016/j.jag.2022.102685>, 2022.
- Jung, J., Maeda, M., Chang, A., Bhandari, M., Ashapure, A., and Landivar-Bowles, J.: The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems, *Current Opinion in Biotechnology*, 70, 15–22, <https://doi.org/10.1016/j.copbio.2020.09.003>, *food Biotechnology - Plant Biotechnology*, 2021.
- Juve, G. and Deelman, E.: Scientific Workflows and Clouds, *XRDS*, 16, 14–18, <https://doi.org/10.1145/1734160.1734166>, 2010.
- Kadri, S., Sboner, A., Sigaras, A., and Roy, S.: Containers in Bioinformatics: Applications, Practical Considerations, and Best Practices in Molecular Pathology, *The Journal of Molecular Diagnostics*, 24, 442–454, <https://doi.org/https://doi.org/10.1016/j.jmoldx.2022.01.006>, 2022.
- Kocon, K., Krämer, M., and Würz, H. M.: Comparison of CNN-based segmentation models for forest type classification, *AGILE: GIScience Series*, 3, 42, <https://doi.org/10.5194/agile-giss-3-42-2022>, 2022.
- Krämer, M.: Efficient Scheduling of Scientific Workflow Actions in the Cloud Based on Required Capabilities, in: *Data Management Technologies and Applications*, edited by Hammoudi, S., Quix, C., and Bernardino, J., pp. 32–55, Springer International Publishing, Cham, 2021.
- Krämer, M., Würz, H. M., and Altenhofen, C.: Executing cyclic scientific workflows in the cloud, *Journal of Cloud Computing*, 10, 1–26, <https://doi.org/10.1186/s13677-021-00229-7>, 2021.
- Kubeflow Authors: Kubeflow, <https://www.kubeflow.org/>, last visited: 2023-02-10, 2023.
- Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z., Fotouhi, F., and Hua, J.: A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution, *IEEE Transactions on Services Computing*, 2, 79–92, <https://doi.org/10.1109/TSC.2009.4>, 2009.
- Linux Foundation: PyTorch, <https://pytorch.org/>, last visited: 2023-02-01, 2023.
- Liu, K., Aida, K., Yokoyama, S., and Masatani, Y.: Flexible container-based computing platform on cloud for scientific workflows, in: *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*, pp. 56–63, IEEE, <https://doi.org/10.1109/ICCCRI.2016.17>, 2016.
- Mattoso, M., Ocaña, K., Horta, F., Dias, J., Ogasawara, E., Silva, V., de Oliveira, D., Costa, F., and Araújo, I.: User-Steering of HPC Workflows: State-of-the-Art and Future Directions, in: *Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, SWEET '13*, Association for Computing Machinery, New York, NY, USA, <https://doi.org/10.1145/2499896.2499900>, 2013.
- Microsoft: Azure AI Platform, <https://azure.microsoft.com/en-us/solutions/ai>, last visited: 2023-02-10, 2023.
- Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., and Fernández-Leal, Á.: Human-in-the-loop machine learning: a state of the art, *Artificial Intelligence Review*, pp. 1–50, <https://doi.org/10.1007/s10462-022-10246-w>, 2022.
- Planet Labs: Planet | Homepage, <https://www.planet.com/>, last visited: 2023-01-19, 2023a.
- Planet Labs: Satellite Imagery Analytics, <https://www.planet.com/products/analytics/>, last visited: 2023-02-15, 2023b.
- Planet Labs: Planetary Variables: Quantifying a Changing World, <https://www.planet.com/products/planetary-variables/>, last visited: 2023-02-15, 2023c.
- Williams, D. L., Goward, S., and Arvidson, T.: Landsat: Yesterday, today, and tomorrow, *Photogrammetric Engineering & Remote Sensing*, 72, 1171–1178, <https://doi.org/10.14358/PERS.72.10.1171>, 2006.