Geosoftware II, WiSe 2024/25
Web Catalogue for user-friendly search and retrieval of machine learning models for EO datacubes

Brian Pondi, Christian Knoth

October 2024

Introduction

Important links

LSF/QISPOS: 148957

LearnWeb: Geosoftware II WS 2024/25

Course goals and schedule

Geosoftware II is set up to be probably one of the most challenging courses of your studies. Students work as a group on a project to solve a complex geoinformatics challenge, which no one has worked on before and where the right path to an acceptable solution is not known, not even by the teachers. The solution will involve choosing the best tool independent of previous programming language experiences.

The students conduct joint group work, forming companies/teams that are the *contractor*, who bids for a project tender published by the *customer*, the teachers. The customer publishes a problem statement and an invitation to bid[1] for solving a relevant problem or business case. The *contractor* provides the managers, architects, designers, and developers of a software system to advance the state-of-the-art in geoinformatics, this year in the context of search and retrieval of deep learning models for spatial (or spatio-temporal) data. Each group bids with their own approach to the project, implements it, and presents it to the customer. Groups are encouraged to name their group and product for easy recognition, including GitHub/GitLab organisation, logo, etc.

In this project, students gain valuable experience for their future jobs as researchers, project managers, software developers etc. The main lessons learned from Geosoftware II should be practical experiences in understanding someone else's problem, designing and implementing a software solution to it, and staying within the project deadlines. The core skills built evolve around the code and programming, especially in *collaborative software development*.

[1] https://en.wikipedia.org/wiki/Invitation_for_bid

Topics for Introductory Presentations

In the first two course sessions, the several topics will be presented by individual students or groups of two. These topics provide background on the project setting and goals. The presenting students are then go-to experts on the respective topic if further questions arise during the course of the semester. The students must provide a handout written in Markdown in form of a pull request to the course's GitHub repository Geosoft2/geosoft2-2024 and may use any suitable means to present their findings.
The topics and allocations are managed in GitHub issues with label 'topic'.

Efforts and grading

9 ECTS correspond to 270 working hours. We allot 23 hours for preparing introductory presentations, attending introductory presentations, and preparing and attending the final presentation, and 2 hours for writing the individual report. That leaves 245 hours over 15 weeks (not counting Christmas holidays) for the project planning and implementation. This gives an average workload of a little bit over two full working days per week during the semester for each student.

What	Est. work load (hrs)	Grade contribution
presentations & report (topic preparation, attendance, final presentation, individual report)	25	20%
planning (bid, revised bid, schedule)	25	15%
implementation (incl. project report, technical documentation, demonstration, regular meetings, project management tasks, etc.)	220	65%

Grading of the implementation is based (a) on an overall grade for the result, the project progression, especially adjusting and reacting to challenges and internal communication and cooperation between colleagues, and (b) individual (code) contributions as shown in the software development platform. Because of the latter, we require each student to contribute under their **own account** and co-author joint commits[2]. A fork & pull development model is highly encouraged because it allows to list the most relevant pull requests in a final **individual report**. This report must present personal *learning achievements*, *role(s) in the team*, and individual *contributions* in a single PDF document (max. 2 pages).

[2] https://github.blog/2018-01-29-commit-together-with-co-authors/

1 Invitation to bid (Lastenheft)

1.1 Project title

Web Catalogue for User-Friendly Search and Retrieval of Machine Learning Models for EO data cubes

2 Problem Statement

The rapid expansion of Earth Observation (EO) data from satellite systems has given rise to vast opportunities for applying machine learning (ML) models to spatiotemporal datasets. However, the accessibility of relevant machine learning models for EO analysis is still a bottleneck for many users. Currently, finding, understanding, and applying models suitable for specific EO tasks—such as land cover classification, anomaly detection, and time-series analysis—can be complex and time-consuming.

To address this, a centralized platform is needed to catalogue machine learning models specifically tailored to EO data analysis. By following the SpatioTemporal Asset Catalog (STAC) standard, and particularly the STAC Machine Learning Model (MLM) extension Charette-Migneault et al. (2024); STAC MLM Extension (2024), this platform will enable users to easily search for and integrate relevant models into their workflows using STAC clients, such as PySTAC or RSTAC.

This platform will allow users to discover, explore, and download metadata and ML models, facilitating integration into Python or R workflows.

3 About the Customer

The customer is a European Union(EU) non-profit research organization specializing in Geospatial Artificial Intelligence(GeoAI) Gao et al. (2023). They have received funding from the EU commission to spearhead democratization of pre-trained models for Earth Observation solutions. Their target audience includes:

- **Machine Learning Engineers/Data scientists**: Looking to share and integrate pre-trained EO models into Python or R workflows using STAC clients like PySTAC or RSTAC.
- **Environmental researchers**: Seeking machine learning models for remote sensing applications (e.g., land use/land cover classification, time-series analysis).
- **Geospatial professionals**: Needing ready-to-use machine learning models to automate geospatial data processing.

The platform will serve as a centralized hub for spatial-temporal machine learning models, emphasizing ease of discovery, exploration, and integration.

4 State-of-the-Art

Existing platforms such as Hugging Face[3] provide a robust framework for discovering and sharing machine learning models but are not specifically tailored to EO data. On the other hand, STAC has emerged as a widely-adopted standard for cataloging

[3] https://huggingface.co/

spatiotemporal assets, such as EO datasets, but the STAC Machine Learning Model (MLM) extension has only recently started to address the need for cataloging machine learning models for EO.

The STAC MLM extension provides a flexible and interoperable way to describe ML models and their metadata. This enables users to efficiently search, retrieve, and apply these models using STAC-compliant tools like PySTAC or RSTAC. However, there is no dedicated platform for browsing and retrieving these models in a user-friendly manner.

5 Project Goals

The goal of this project is to create a web-based catalogue, similar in functionality to Hugging Face, but focused on machine learning models designed for Earth Observation (EO) applications. The platform will:

1. **Catalog ML Models**: Provide a searchable, filterable catalogue of ML models for EO, with metadata following the STAC MLM extension.
2. **Enable Model Integration**: Allow users to integrate these models into their workflows via STAC clients (e.g., PySTAC, RSTAC).
3. **Metadata Upload and Download**: Support uploading and downloading of spatiotemporal metadata, enabling users to fetch models in standard formats that can be easily integrated into their Python or R workflows.
4. **Community Contribution**: Allow users to upload their own models, contributing to the platform's repository following the STAC MLM extension.

6 Requirements

The requirements of this project are to develop software to run a web-based catalogue for machine learning models focused on EO applications that includes:

- a front-end instance that provides the user interface including, amongst others, overviews/summaries of model metadata, an interface for uploading and downloading metadata, and the option to search for models by specific requirements (e.g., EO task, data type)
- a back-end instance that stores and manages the model metadata following the STAC MLM extension specification

The back-end instance with some exemplary data and the front-end instance should be running at time of project delivery; one or multiple Dockerfile(s) that, when built and started, run(s) an instance of the service shall be made available.

6.1 Functional Requirements

The following functional requirements extend and clarify the above overall requirements, but they may not cover all features needed to realize the project goals.

6.1.1 Model Catalogue and Search

STAC-Compliant Model Metadata:

- The catalogue will store and manage model metadata. It will enforce that the metadata follow the STAC MLM extension specification, including information about:
  (a) Model task (e.g., classification, regression, anomaly detection)
  (b) Temporal and spatial applicability (e.g., region of interest, date ranges)
  (c) Data type (e.g., Sentinel-2, Landsat) that the model was trained on.
  (d) Input/output requirements (e.g., resolution, number of bands, etc.)

Search and Filtering:

- Allow users to search models by task (e.g., classification, regression)
- Allow users to filter by geographic region, data type (e.g., Sentinel-2), temporal range, and/or cloud coverage
- **BONUS**: include additional filtering options based on training data coverage

Model Exploration:

- Provide detailed metadata pages for each model, including:
  (a) Model description and architecture
  (b) Intended use cases and limitations
  (c) Hyperparameters, input/output data formats, and any additional notes on model usage

6.1.2 Model Upload and Metadata Handling

Model Upload Interface:

- Users can upload new machine learning model metadata, ensuring that it complies with the STAC MLM extension
- Provide a step-by-step guide for preparing the necessary metadata files, with validation to ensure compliance

STAC Metadata Generation:

- Support automatic generation of STAC-compliant metadata for machine learning models during the upload process, ensuring the correct structure and fields are in place

6.1.3 Integration with STAC Clients

STAC-Client Support:

- Ensure compatibility with STAC clients like PySTAC (Python) and/or RSTAC (R)
- Provide clear instructions and code snippets to help users fetch models and metadata directly into their Python or R workflows

Model Fetching:

- Allow users to download the STAC metadata for any machine learning model
- The metadata will include links to the model files (e.g., stored on cloud storage) that users can download or fetch using STAC-compliant clients

Demonstrate Client Integration:

- Extend either RSTAC and/or PySTAC client software to be able to search and retrieve a model through the developed catalogue

6.1.4 Data Download

STAC Metadata Download:

- Allow users to download the model's STAC metadata, which can then be integrated into their workflows using PySTAC or RSTAC

Model File Download:

- Provide links to download the model file (e.g., in ONNX, TensorFlow, or PyTorch formats) associated with the metadata

6.1.5 Community Contributions

Upload and Contribution:

- Allow users to contribute models that follow the STAC MLM extension, fostering a community-driven repository of machine learning models for EO applications
- **BONUS**: Implement a user management system so that only registered and logged-in users can upload models or model metadata

6.1.6 Visualization

Model Visualization:

- Provide visual previews (e.g., model input data, geographic areas covered) to help users quickly assess model suitability

6.2 Non-Functional Requirements

6.2.1 Training and Demonstration

- **Documentation**: Provide comprehensive documentation on how to use the platform and integrate models into workflows
- **Tutorials**: Include tutorials and example notebooks for Python (PySTAC) and R (RSTAC), showcasing how to query and apply models

6.2.2 User Friendliness

- **Intuitive UI**: Provide a clean and easy-to-use web interface, similar to Hugging Face, where users can search for models, view detailed metadata, and download STAC-compliant files. Common practices for web-based user interfaces and interaction paradigms should be applied. The system must be tested and fully functional with browsers representing at least 80% of current users.
- **Color-Blind Accessibility**: Ensure that the platform is accessible to users with color vision deficiencies, using appropriate color schemes and UX design.
- **BONUS**: Include help pages and tutorials on using STAC clients to fetch and apply models within Python or R workflows.

6.2.3 Performance

In general, a user interaction must result in a display change within 1 second to allow users to stay focused on their current train of thought; complex page contents may be loaded asynchronously; interactive visualisations must react within 0.1 seconds to give a user the impression of direct manipulation.[4]

6.2.4 Maintainability

- **Open Source**: The platform should be published under an OSI-approved open-source license. The contractor must ensure license compatibility of other used or extended software.
- **Documentation**: Documentation on building, installing, testing, and configuring the system must be provided in Markdown-formatted documentation files using the appropriate markup for lists, links, etc. The documentation must be rendered to an online website.
- BONUS: **Continuous Integration (CI)**: Set up automated pipeline for updating Docker images of the platform with GitHub actions.

6.2.5 Deployment

- **Doker**: Ensure the platform is deployable via Docker, enabling easy setup on cloud environments such as AWS or Google Cloud.
- **Docker Hub**: Used images must be based on Dockerfiles published as part of the source code, and all base images should be hosted on Docker Hub (or comparable).

6.2.6 Project Management

- The customer will be given access to an online task management system to track the progress of the project.

6.2.7 BONUS: Suggestions on specifications to catalog Training Data

- The customer would like to know your suggestions on the ideal approach to catalog Spatio-Temporal Training Data.

[4]Source: http://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/

7 Deliverables and deadlines

Bid The bid (Pflichtenheft[5]) must be submitted to the customer via UNIM's Learnweb no later than **October 29, 2024, 12:00 CET**. It must comprise an implementation plan for all requirements of the invitation to bid (Lastenheft) as well as a schedule for the implementation as a single PDF document. The bid must include *user stories* realising all requirements. The bid document may be English or German. Its content should follow common standards for content and structure of bids.

Changes The bid is a binding agreement between customer and contractor. All changes to the bid after first acceptance require a written confirmation (email) by the other party. The contractor may direct *change requests* to the customer via email to both `brian.pondi@uni-muenster.de` and `christian.knoth@uni-muenster.de`. A change request includes a short summary of the changes in the email body and an updated version of the bid as an attachment.

Final delivery The final delivery must be submitted on **the day before** the final presentation on **January 29, 2025**, and contain the following items:

1. project report (in English or German)
   (a) a single PDF document submitted via Learnweb
   (b) cover page contents: project title, team name (optional), references to all parts of the delivery (code, documentation, etc.)
   (c) screen-shots of all relevant UI components and workflow steps
   (d) description how all requirements/user stories from the bid were fulfilled
2. commented source code (comments in English) in a single code repository on ZIVGitLab, GitLab.com, or GitHub.com; delivery in more than one repository must be formally accepted by the customer
3. ready-to-use Docker images in a public registry (with docker-compose configuration, if multiple mages are used) and Dockerfiles
4. installation and maintenance documentation (English, online website, source may be included in repository or in a separate repository)

Acceptance criteria The acceptance criteria encompass the fulfilment of all functional and non-functional requirements as described in the accepted bid, and the complete and on-time submission of all deliverables.

References

Charette-Migneault, F., Avery, R., Pondi, B., Omojola, J., Vaccari, S., Membari, P., Peressutti, D., Yu, J., and Sundwall, J. (2024). Machine learning model specification for cataloging spatio-temporal models (demo paper). In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Searching and Mining Large Collections of Geospatial Data (GeoSearch'24)*, page 4 pages, New York, NY, USA. ACM.

[5] https://de.wikipedia.org/wiki/Pflichtenheft, https://wiki.induux.de/Pflichtenheft

Gao, S., Hu, Y., and Li, W., editors (2023). *Handbook of Geospatial Artificial Intelligence*. CRC Press, 1st edition.

STAC MLM Extension (2024). Stac mlm: Spatiotemporal asset catalog machine learning model extension. https://github.com/stac-extensions/mlm. Accessed: 2024-10-14.