FAIRICUBE – F.A.I.R. INFORMATION CUBES

WP4 Share
D4.5 Apps to support community collaboration platform

Deliverable Lead: EOX
Deliverable due date: 30/06/2024

Version: 2.0
2024-06-22

Document Control Page

Document Control Page	
Title	
Creator	EOX
Description	Based on the collected requirements, new apps, based on existing Open-Source software, will be configured and offered for deployment in users' workspaces on the FAIRiCUBE HUB. The apps themselves will be released as Open-Source on GitHub again. Care will be taken to assure close integration of ML capabilities with the data resources being provided.
Publisher	"FAIRICUBE – F.A.I.R. information cubes" Consortium
Contributors	All partners
Date of delivery	2024-06-22
Type	Text
Language	EN-GB
Rights	Copyright "FAIRICUBE – F.A.I.R. information cubes"
Audience	☒ Public ☐ Confidential ☐ Classified
Status	☐ In Progress ☐ For Review ☒ For Approval ☐ Approved

Revision History

Version	Date	Modified by	Comments
0.1	24/11/2023	Christian Schiler (EOX)	Initial release
0.2	27/11/2023	Christian Schiler (EOX)	Added Apache Superset description
0.3	07/12/2023	Christian Schiler (EOX)	Added Requirements collected at Meeting 05/12/2023
1.0	15/12/2023	Jaume Targa	Internal review
1.1	28/05/2024	Christian Schiler (EOX)	Restructuring Chapters Add introduction of collaborative setup Description of collaborative tools added chapter anout rasdaman ML-tools & UDF List of Requirements placed in Annex
2.0	22/06/2024	Jaume Targa, Stefan Jetschny	Review and format checking

Disclaimer

Table of Contents

List of Figures

List of Tables

1 Introduction

This is the deliverable "D4.5 Apps to support community collaboration platform" of the FAIRiCUBE project, i.e., the FAIRiCUBE integrated datacube platform. It describes the process to collect the requirements for machine learning (ML) toolkits from the use case partners. Based on the collected requirements, new apps, based on existing Open-Source software, will be configured for deployment in users' workspaces on the FAIRiCUBE HUB. The apps themselves will be released as Open-Source on GitHub again. Care will be taken to ensure close integration of ML capabilities with the data resources being provided.

Ideally, classical operations on the Analysis Ready Data Cubes (ARDC) could be extended by Machine Learning applications to sustain interoperability. However, there is a conflict between the nature of remotely sensed data, the structure of the ARDCs and the requirements for meaningful Machine Learning applications which need to be addressed:

1. Sampling the Earth naturally leads to an uneven distribution of data points as a result of its spherical shape. This phenomenon is reinforced by data gaps due to e.g., satellite trajectories or cloud cover. Hence, there is no uniform data distribution across the chunks of the ARDC provided.
2. Remotely sensed data tends to be auto-correlated within (neighbouring) chunks as data points which are in close spatio-temporal vicinity are naturally characterized by a low variance.

Therefore, it is mandatory to enable Machine Learning that respects the basic principles of geo-data way beyond naive applications of Machine Learning in the Earth system context. To avoid auto-correlation during the training phase of the model, we propose that data sampling should rather be guided by a block sampling strategy instead of random sampling, which accounts for challenge 1. and 2. and enables efficient and more meaningful ML applications on ARDCs. Generally speaking, data blocks can be masks of any kind (e.g., data thresholds, temporally restricted or spatially shaped). For example, blocks that are rectangular shaped varying in size and amount of data points can be used.

1.1 Setup for Collaboration

Although not directly an App it has to be mentioned here that the FAIRiCUBE Lab does provide the direct opportunity to share data and code within the FAIRiCUBE community. All Use Cases utilizing the JupyterLab instance in FAIRiCUBE have access to an use case specific storage as well as to a FAIRiCUBE-wide common storage space. These storage areas exist beside the commonly available space to share information via the available FAIRiCUBE GitHub repositories. Furthermore, the implemented Jupyter Notebook Viewer (for details see Chapter 2.4) represents a tool which renders Jupyter notebooks as static web pages. These are then easily shared. For the collaborative documentation of processes, code and know-how within FAIRiCUBE, a read-the-docs instance is setup to be used and filled with information by all users (for details see Section 2.5).

2 General Applications provided

2.1 Apache Superset

Based on an already issued special request for a tool to support easy data visualization by UC2 the Apache Superset has been deployed and made available for UC-2 users. Apache Superset is an open-source software application for data exploration and data visualization able to handle data at a petabyte scale (big data). It makes it easy to explore your data, using either a simple no-code viz builder or a state-of-the-art SQL IDE. In addition, Superset can connect to any SQL-based databases including modern cloud-native databases and engines. It ships with 40+ pre-installed visualization types that make it easy for users of all skill sets to explore and visualize their data, from simple line charts to highly detailed geospatial charts. Plug-in architecture makes it easy to build custom visualizations.

2.2 Grafana

Grafana is a multi-platform, open-source analytics and interactive visualization web application. The tool is very flexible and provides a multitude of APIs to allow various data sources to be utilized and integrated. In addition, it provides many possibilities to present the data and also integrate additional external resources (e.g. Wikipedia) directly or scrape their content to present it in a specific way. It can produce charts, graphs, and alerts for the web when connected to supported data sources. As a visualization tool, Grafana can be used as a component in an application stack together with tools providing the respective static or dynamic input. Complex dashboards can be built by end users, with the aid of interactive query builders.

2.3 EO-Dashboard

The Earth Observing Dashboard (EO-Dash) has been designed and developed by EOX in the course of the coronavirus pandemic as an effort of the three space agencies, ESA, JAXA, and NASA, to strengthen our global understanding of global environmental changes and other societal challenges impacting our planet. The initial Race-Dashboard[1], was first developed to provide information observable from Space about the Covid-19 Crisis e.g. the reduced discharge of CO2 due to traffic limitations, monitoring of empty parking lost etc. However, EO-Dash has thereafter evolved further and has been used for various purposes due to its flexibility and ease of setup and use. EODash is intended to handle and present data provided by users (third parties) who want to tell a story about a distinct event or occurrence. EO-Dash demonstrates how the use of Earth observation data can help shed new lights on societal and economic changes taking place.

2.4 Jupyter Notebook Viewer

Jupyter Notebooks extend the console-based approach to interactive computing with a web-based application, with which the entire process can be recorded: from developing and executing the code to documenting and presenting the results. The Notebook Viewer renders the interactive notebooks and

1 https://race.esa.int/

creates static web pages. These static web pages can then easily made available to other users for reading without the danger that another user will destroy/change the notebook by accident. However, since notebooks in FAIRiCUBE are stored in GitHub all offered notebooks are fully accessible for usage by others.

2.5 Read-the-Docs

Read-the-Docs is a simple tool for building, hosting and sharing documentation. It is based on Mark-down language and therefore doesn't need special training. The documents can be hosted easily in GitHub/GitLab and when connected to the read-the-docs site will be built and deployed automatically after every change. All users which have access to the GitHub/GitLab location can add and edit the documentation. Texts, images, code, and even Jupyter Notebooks can all be included, managed and presented in a read-the-docs documentation.

2.6 Common tools developed in FAIRiCUBE

A whole set of notebooks has been and is actively developed during the FAIRiCUBE project. These tools are made available by the via GitHub and can currently be used by other FAIRiCUBE members. Since this is a ongoing activity a listing of available tools is not provided here. However, a more comprehensive description of the development process can be found in the deliverable "D3.3_Processing and ML applications.docx". In general, the descriptions of available tools and their usage can be found in the FAIRiCUBE Read-the-Docs collection and are provided by the respective developers. Also, the FAIRiCUBE catalog for a/p resources are a valuable tool to search and find such common tools developed in FAIRiCUBE.

3 ML-Toolkits available on the FAIRiCUBE Hub

Each Use Case team decides which apps shall be made available. At the time of writing two apps have been made available, currently associated with UC2. Two of these apps are MLflow and TensorBoard, for which a short introduction is provided in the next sub-chapters. An additional option to use Machine learning is setup on rasdaman (3.4) using User Defined Functions (UDFs) to provide the PyTorch framework to users. Not ML but also a helpful tool is DVC, a Data Version Control system, for ML modelling and collaborative data management like ML artefacts, is also described here in this section.

3.1 MLflow

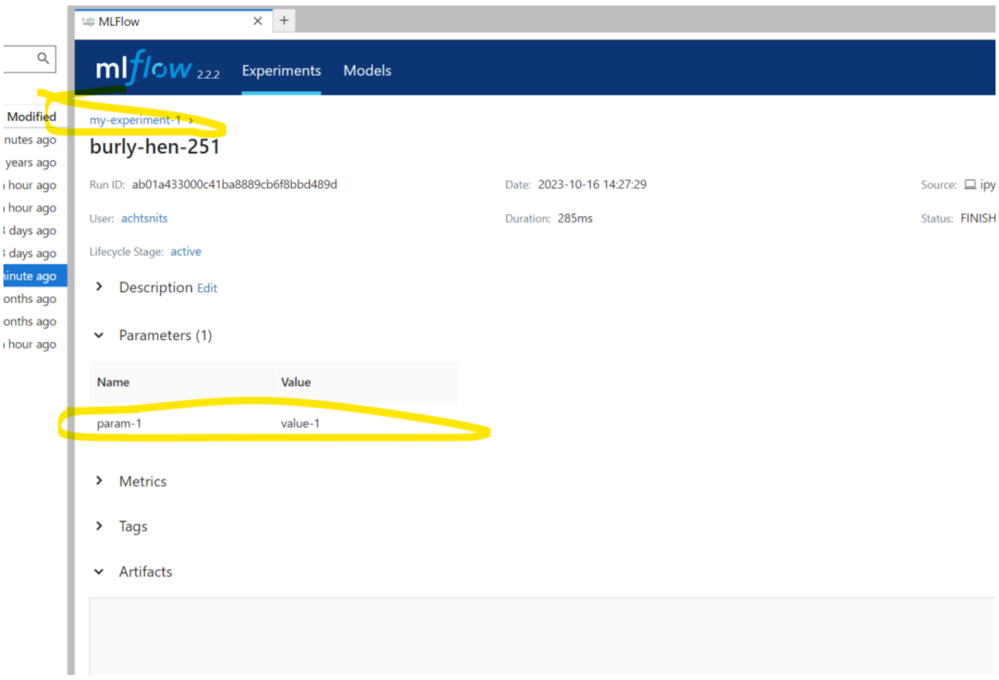
MLflow is an open-source platform for managing the end-to-end machine learning life cycle. It allows the user to track experiments, package code into reproducible runs, and share and deploy models. MLflow can be incorporated into Jupyter notebooks or other code and supports multiple programming languages. It is widely used in industry and academia and is constantly evolving to support the latest trends and technologies in the field of machine learning. MLflow supports the MLOps pipelines particularly to log and evaluate experiment runs as well as to store models in a registry. Persistent MLflow deployments are made available on the team level to allow each team member to compare their experiments with those of the other team members and to use the trained models of others. At a high level, MLflow consists of four main components: tracking, projects, models, and registry. All components can be accessed via Python code in the FAIRiCUBE Lab.

MLflow Tracking: The MLflow tracking component allows users to log and track training parameters, code, and output metrics from their machine learning experiments. It provides an API for Python, R, and other languages, as well as a UI for visualising experiments and comparing different runs. The tracking server can store data in various backends, including a local file system, an Amazon S3 bucket, or a PostgreSQL database. MLflow Tracking uses the concept of runs, which are executions of some piece of data science code, e.g., training of models. MLflow Tracking supports auto-logging for many classic libraries such as TensorFlow, Scikit-Learn, Spark, or PyTorch, but manual logging is available in other cases.

MLflow Projects: The MLflow projects component provides a standard format for packaging and distributing machine learning code, including dependencies, in a reproducible way. Projects can be run locally or in a cluster, and MLflow can manage the environment and dependencies for each run. In addition, the Projects component includes an API and command-line tools for running projects, making it possible to chain together projects into workflows. In the MLproject file, it is possible to define the software environment and entry points with parameters to define workflow.

MLflow Models: The models component allows users to easily package models in a standard format for deployment. Models can be exported in multiple formats, including TensorFlow, PyTorch, and ONNX, and can be deployed using a variety of tools, including Docker, Kubernetes, and Amazon SageMaker. It is also possible to access models in standard ways such as REST API. The Model API allows saving, loading, and logging of the model as well as adding different flavours.

MLflow Model Registry: The MLflow Model Registry component allows users to store, manage, and deploy models in a central repository. Models can be versioned, and access can be controlled using role-based access control. The Model Registry works both in UI and API versions.



my-experiment-1 >

burly-hen-251

Run ID: ab01a433000c41ba8889cb6f8bbd489d
Date: 2023-10-16 14:27:29
Source: ipy
User: achtsnits
Duration: 285ms
Status: FINISH
Lifecycle Stage: active

› Description Edit

⌄ Parameters (1)

Name	Value
param-1	value-1

› Metrics

› Tags

⌄ Artifacts

Figure 1: MLflow - Experiments tracking interface

3.2 TensorBoard

To support model evaluation during training, the FAIRiCUBE Hub processing environment is also extended by a **TensorBoard** to support the tracking of individual experiments and training runs. This tool can be used with PyTorch and TensorFlow, and it provides a state-of-the-art toolset for data scientists to inspect the tuning and training process and compare metrics. In machine learning, to improve something you often need to be able to measure it. TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower dimensional space, and much more. TensorBoard provides the visualization and tooling needed for machine learning experimentation:

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Projecting embeddings to a lower dimensional space
- Displaying images, text, and audio data
- Profiling TensorFlow programs

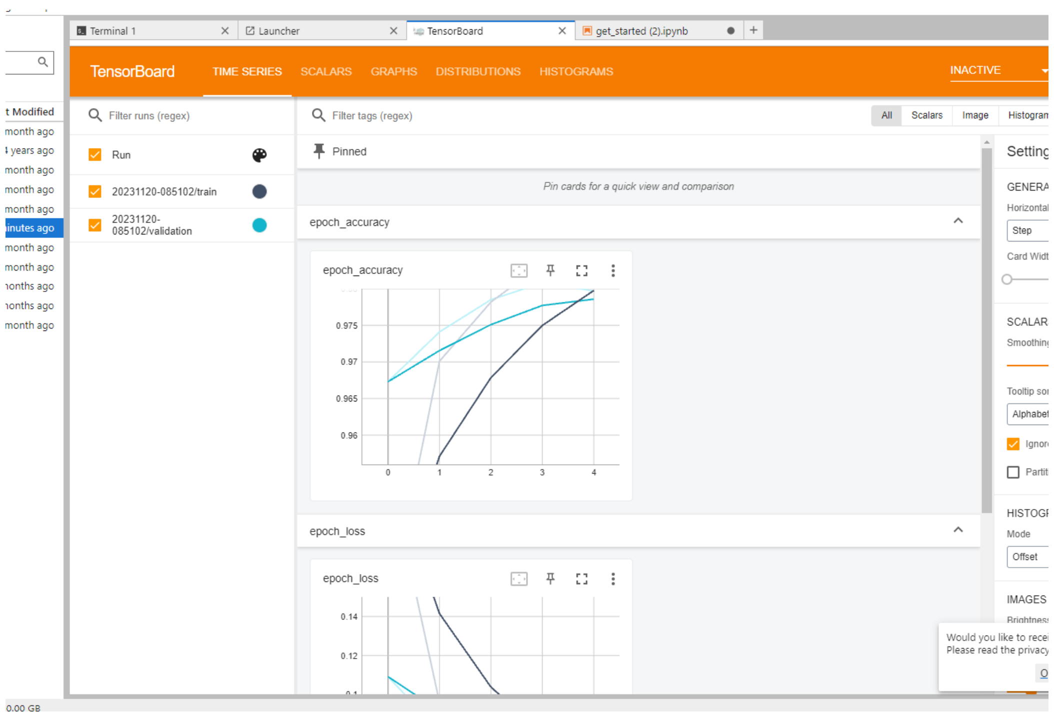


Figure 2: TensorBoard – Time series evaluation

3.3 DVC

DVC[2] is a Data Version Control system for collaborative data management like ML artefacts. It is a free and open-source, platform-agnostic version system for data, machine learning models, and experiments. It is designed to make ML models shareable, experiments reproducible, and to track versions of models, data, and pipelines. DVC works on top of Git repositories and cloud storage.
DVC's features can be divided into three categories: data management, pipelines, and experiment tracking.

Data management: Data and model versioning is the base layer of DVC for large files, datasets, and machine learning models. It allows the use of a standard Git workflow, but without the need to store those files in the repository. Large files, directories and ML models are replaced with small meta-files, which in turn point to the original data. Data is stored separately, allowing data scientists to transfer large datasets or share a model with others.

Pipelines: DVC provides a mechanism to define and execute pipelines. Pipelines represent the process of building ML datasets and models, from how data is pre-processed to how models are trained and evaluated. Pipelines can also be used to deploy models into production environments. DVC pipeline is focused on the experimentation phase of the ML process. Users can run multiple copies of a DVC pipeline by cloning a Git repository with the pipeline or running ML experiments. They can also record the workflow as a pipeline and reproduce it in the future.

2 https://dvc.org

Experiment tracking: tracking allows developers to explore, iterate and compare different machine learning experiments. Each experiment represents a variation of a data science project defined by changes in the workspace. Experiments maintain a link to the commit in the current branch (Git HEAD) as their parent or baseline. However, they do not form part of the regular Git tree (unless they are made persistent). This stops temporary commits and branches from overflowing a user's repository.

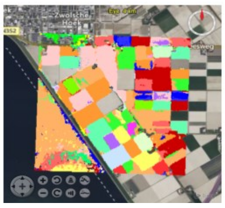
3.4 rasdaman ML tools

The models created by the Use Cases can be run directly inside the rasdaman server. Via the extensibility feature mentioned above the PyTorch framework (upon recommendation by WER) has been integrated into rasdaman so that now ML models build with PyTorch can be activated from a query, together with the input data for the inference.

Implementation was done using the rasdaman extensibility mechanism of User Defined Functions (UDFs). This feature allows dynamic loading of external code at query time, where this functionality can be invoked seamlessly integrated as if it were a function of the query language. To this end, some adapter code has to be written, originally in the only supported language, C++. For PyTorch, the initial idea was to use torch directly which is implemented in C++ and, therefore, allows for a direct coupling. It turned out, however, that PyTorch is not just a python wrapper but performs own data preparation, and so the torch coupling delivered wrong results. Consequently, it was necessary to add Python as an additional language for UDFs. This was accomplished in the project, leading to a dual result:

- Via a UDF function nn.predict(), models that have been registered with rasdaman can be passed to PyTorch as part of the query, and the result can be processed further in the query or delivered directly.
- In general, any Python code can act as a UDF in rasdaman. As Python is so common among data scientists such a Python coupling provides an additional advantage for them.

Figure 3 shows a sample model application to a region in the Netherlands using a model provided by WER. This close integration allows a very simple, interactive application of models to any datacube, any region, and any time – if desired, combined with the full set of WCPS operators. Current status is that models, in order to be available in queries, have to be provided on the server in a designated directory by someone authorized (such as WER). In future, it is planned to generalize this to also allow passing of models on the fly, as part of a query. Further, models should migrate into the database and get attributes which allow users to meaningfully select their model of choice. One such criterion is the region (in space and time) of applicability of a particular model (minimizing the risk of applying a machine learning model on out-of-distribution data).



```
for $s2 in (Sentinel_2),
    $m in (CropModel)
return
    encode( nn.predict( $s2[…], $m ), "tiff" )
```

Figure 3: Sample model application as a query result (left) and schematic WCPS query generating it.

4 Collection of User Requirements

Although the provided listing of User Requirements is, from a technical point of view, very informal, it provides ample information for the supplier/operator (EOX) to allow the evaluation and feasibility of the implementation of the respective tool in the FAIRiCUBE-Lab. Since all tools will be run encapsulated (e.g. Docker container) in the cloud, a detailed technical evaluation of each tool is not required from the supplier/operator side. A dedicated feasibility study for each software tool is therefore also not deemed necessary. Highly specialised software (e.g. like "vcftools" for genetic analysis) are simply placed inside container and then provided to the user.

However, to provide the best Toolkits to the users to best support their needs, a meeting for all partners was arranged. At the beginning of this meeting an introductory lecture, on the already available MLToolkits (MLflow and TensorBoard), was provided by EOX. Thereafter, a discussion on further requirements was initiated and the various views and wishes expressed by the use case partners have been collected and harmonized. During this discussion also non-ML requirements for needed tools have been collected and are listed below.

While some tools are very Use Case specific (e.g. vcftools, for working with complex genetic variation data), others are of general purpose. As an example, Grafana can be used for the presentations of measurement and graphics of all kinds of information. Collaboration inside UC teams, as well as across UC teams is fostered by either direct information exchange between the UC's (e.g. commonly shared folders and buckets) and/or by providing descriptions of the developed tools via the Read-the-Docs collaboration platform or via the Knowledge Base. The respective code of these developed tools is commonly available via the GitHub repository (e.g. https://github.com/FAIRiCUBE/common-code**Error! Hyperlink reference not valid.**

Request for the provisioning of additional tools via the FAIRiCUBE Hub can always be submitted to the operations team using GitHub Issues (https://github.com/FAIRiCUBE/FAIRiCUBE-Hub-issue-tracker). The listing of these requirements asked for by the respective UC's is provided in Table 1.

Use Case	Requirement	Note
UC-2	Enable MLflow model registry • MLflow & PyTorch enabled • installed and activated DoWhy library	implemented
UC-2/ UC-3	Pipelines like "Apache airflow" (ETL like) – or "prefect" • also include cmd-line tools • used to ingest/create datacubes	implemented
UC-3	Need vcftools - biological tool https://github.com/vcftools/vcftools (C program, https://en.wikipedia.org/wiki/Variant_Call_Format, for genetic data) used by NHM • used by UC's directly, no setup in FAIRiCUBE Hub was necessary	implemented
UC-2	needs JDK in order to run: https://biodiversityinformatics.amnh.org/open_source/maxent/ • it is sufficient to run from command line (headless execution possibility)	pending
UC-1-	needs tool(s) for graphical presentation and sharing of results • provided graphical tools are: Apache superset, Grafana, eo-dashboard	implemented

WP-4	tool for collaborative documentation needed • Provided tool: Read-the-Docs in combination with GitHub	implemented
Mgmt.	user management tool (GUI) is needed	implementation in progress
Mgmt.	get information about Costs from Amazon to be seen also by users • start with sharing the monitoring dashboard	implemented

Table 1: List of collected requirements for additional applications requested by the UCs