

TEAMING.AI 7TH PRESS RELEASE

⟨ teamıng_ai

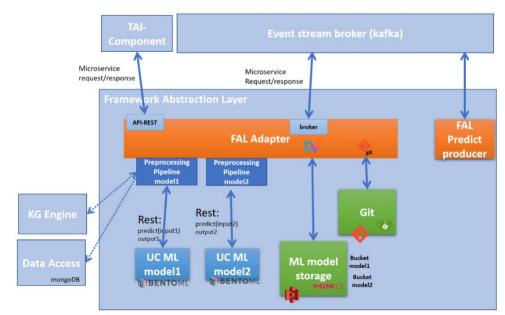
October 2023

Machine Learning Framework Abstraction Layer

The Research Centre Ideko, specialist in advanced manufacturing, leads the integration work package and the open-source project task within the project. It is responsible of developing the framework abstraction layer, whose purpose is to facilitate the integration of machine learning processes within the Teaming AI platform. The main purpose of the framework abstraction layer (FAL) within this project is to facilitate the deployment of machine learning models by providing a consistent and simplified interface.

The machine learning framework abstraction layer acts as a layer of abstraction over the implementation details of the underlying machine learning algorithms, facilitating the deployment of the models. It provides a range of functions and utilities that simplify common tasks such as data loading and pre-processing, model selection and inference.

Within the scope of the project the framework abstraction layer facilitates machine learning model portability across different environments. By providing a uniform abstraction layer, models trained in one framework can be easily used in other machine learning frameworks without significant changes to the code. Framework abstraction layer also includes tracking features that facilitate the monitoring and analysis of machine learning models throughout their lifecycle. These tracking features allow data scientists to keep track of important metadata and metrics associated with their models.



The FAL is composed by different components, shown in Figure 1, where the core components are FAL adapter, ML (machine learning) model storage and FAL Predict Producer.

Figure 1: Framework Abstraction Layer Diagram



FAL Adapter

The FAL Adapter is the component which handles microservice requests. The microservices provided by FAL adapter are the following:

- DeployModel(modelName,version,objectName,objectType). Deploys a version of a ML model in a Docker container.
- Predict(modelName, version,data). Make a prediction using a deployed model.
- CreateModelVersion(modelName, version) . Creates a model version.
- AddObject(modelName,version,objectName, object). Add an object to the repository or updates it.
- RetreiveObject(model name, version, objectName). Retrieve a version of the object.
- ListObjects (modelName, version). List all the objects stored for a model.
- DeleteObject (modelName,version, objectName,version). Remove the object of a particular model/version.
- DeleteModelVersion(modelName, version). Deletes a model version.
- ListModelVersions.(model name): List versions of a model

ML Model Storage

The ML model storage is the backend dedicated to store machine learning models and datasets. It is responsible to manage and organize the data required for the deployment of machine learning models. This component provides the following features:

- Data Storage and Retrieval: provides a reliable and scalable way to store large volumes of data, including trained models, training datasets, model checkpoints, and other supporting files.
- Data Versioning and Management: allows to maintain different versions of datasets and model artifacts.
- Data Sharing and Collaboration: facilitates data sharing and collaboration among different teams or individuals working on the same project.

This component uses DVC (Data Version Control) in conjunction with Object storage engine (MinIO, S3, etc) and version control engines (Github, Gitea) to enhance the versioning and management of machine learning models and associated datasets.

By combining DVC, MinIO, and Git, we can achieve comprehensive version control and management for both data and code in machine learning projects. DVC ensures efficient versioning and sharing of large datasets stored in MinIO, while Git handles the versioning and collaboration of code and configuration files. This integration promotes reproducibility, traceability, and collaboration in the machine learning workflow, enabling teams to effectively manage and track changes to both data and code throughout the development and experimentation process.

FAL Predict Producer

This component facilitates the automated creation of prediction microservice requests through the use of a designated producer topic. It generates predictions automatically whenever new input from Data Producers is published, eliminating the need for Teaming engine orchestration when it is not required.



For additional information please contact

Project Coordinator: Software Competence Center Hagenberg GMBH (SCCH) Bernhard Moser <u>bernhard.moser@scch.at</u>, Mario Pichler <u>Mario.Pichler@scch.at</u> Communication and Dissemination Manager: Core Innovation Maria Lentoudi <u>mlentoudi@core-innovation.com</u>

Connect with us



