

AI4 |  eosC

# DEEPaaS API Community API for AI in the EOSC

Álvaro López García ([aloga@ifca.unican.es](mailto:aloga@ifca.unican.es))  
Ignacio Heredia Cacha ([iheredia@ifca.unican.es](mailto:iheredia@ifca.unican.es))



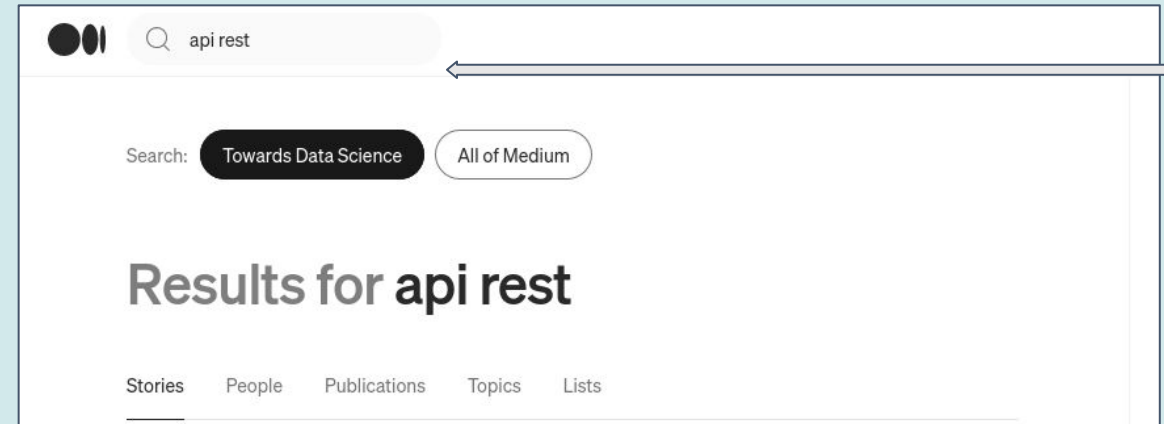


# What is DEEPaaS

- Python API to expose AI/ML model functionality
- Why?
  - Users (data scientists) do not need to program REST APIs
  - Provide homogeneous entry points for AI applications and services
- Easy to...
  - Integrate with user code
  - Integrate with AAI systems (thanks to FLAAT)
  - Deploy and scale behind load balancers

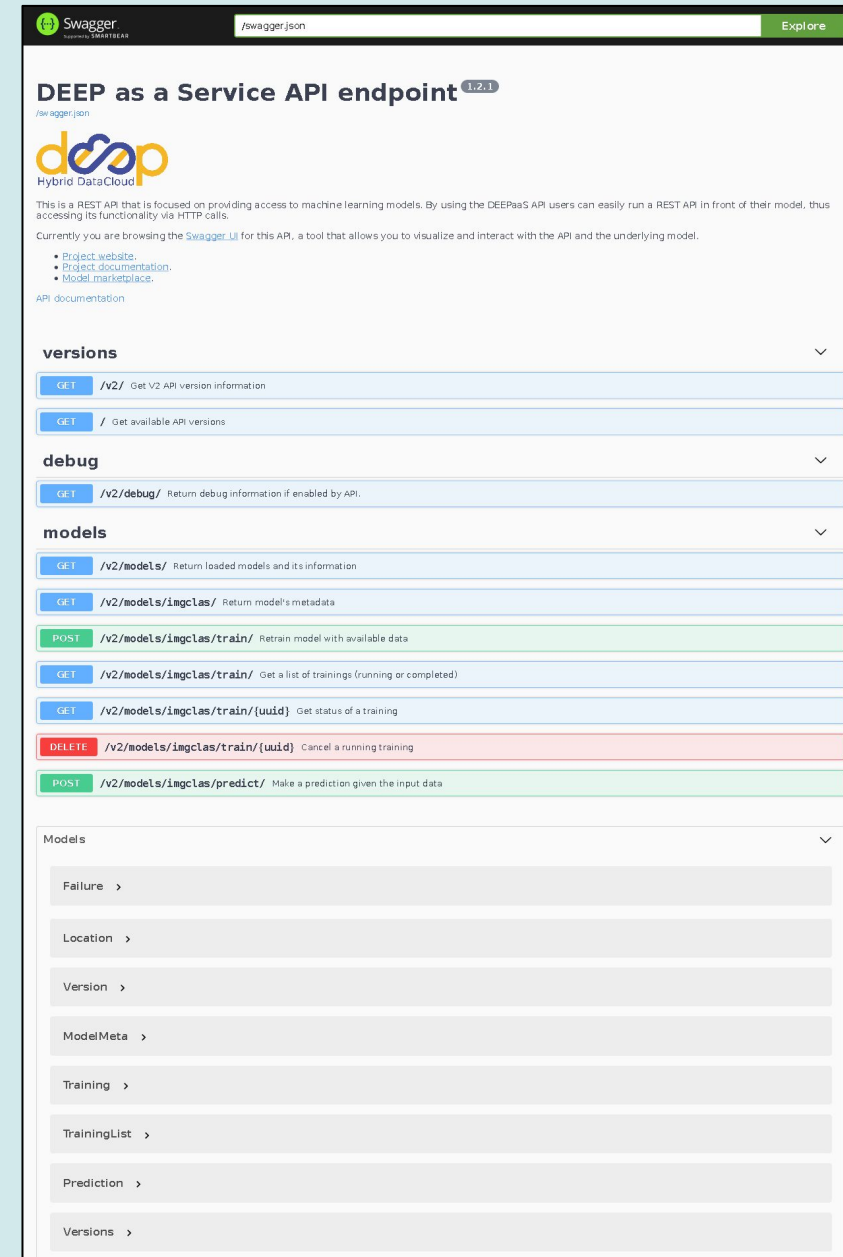
# What is DEEPaaS

- Python API to expose AI/ML model functionality
- Why?
  - Users (data scientists) do not need to program REST APIs
  - Provide homogeneous entry points for AI applications and services
- Easy to...
  - Integrate with user code
  - Integrate with AAI systems (thanks to FLAAT)
  - Deploy and scale behind load balancers



# DEEPaaS functionality

- **train endpoint**
  - Allows users to perform and monitor training
- **predict endpoint**
  - Provides inference functionality
- **debug endpoint**
  - Provides debug information about model behaviour
- **Running in batch? We got you covered!**
  - Wrapper scripts for local inference and prediction



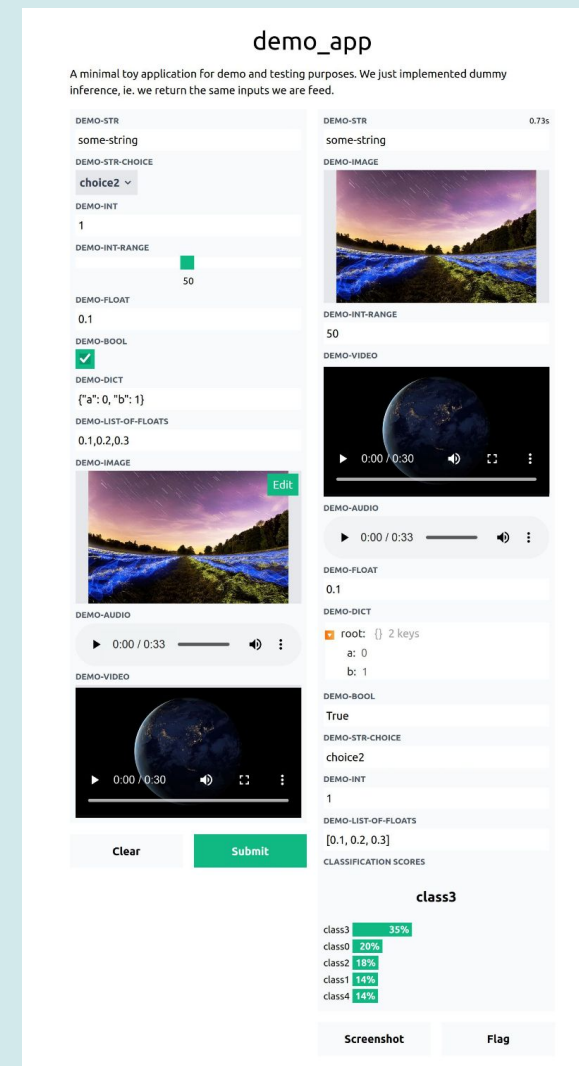
The screenshot shows the Swagger UI for the DEEPaaS API endpoint. The page title is "DEEP as a Service API endpoint" with a version indicator "1.2.1". The logo for "deop Hybrid DataCloud" is visible. The page describes the API as a REST API for machine learning models. It lists several endpoints under different categories:

- versions**
  - GET /v2/ Get V2 API version information
  - GET / Get available API versions
- debug**
  - GET /v2/debug/ Return debug information if enabled by API.
- models**
  - GET /v2/models/ Return loaded models and its information
  - GET /v2/models/imgclas/ Return model's metadata
  - POST /v2/models/imgclas/train/ Retrain model with available data
  - GET /v2/models/imgclas/train/ Get a list of trainings (running or completed)
  - GET /v2/models/imgclas/train/{uuid} Get status of a training
  - DELETE /v2/models/imgclas/train/{uuid} Cancel a running training
  - POST /v2/models/imgclas/predict/ Make a prediction given the input data

At the bottom, there is a "Models" section with a list of expandable items: Failure, Location, Version, ModelMeta, Training, TrainingList, Prediction, and Versions.

# DEEPaaS API functionality

- Need a (nice) user interface apart from the API?
- DEEPaaS together with DEEPaaS UI (beta)
  - Built on the Gradio UI package
  - [https://github.com/deephdc/deepaas\\_ui](https://github.com/deephdc/deepaas_ui)



The screenshot displays the 'demo\_app' interface, which is a minimal toy application for demo and testing purposes. It features a grid of input fields for various data types, including strings, choices, integers, ranges, floats, booleans, dictionaries, lists, and images. The interface also includes a video player and an audio player. At the bottom, there is a 'Submit' button and a 'Classification Scores' section showing a bar chart for 'class3' with a score of 35%.

demo\_app

A minimal toy application for demo and testing purposes. We just implemented dummy inference, ie. we return the same inputs we are feed.

DEMO-STR  
some-string

DEMO-STR-CHOICE  
choice2 ▾

DEMO-INT  
1

DEMO-INT-RANGE  
50

DEMO-FLOAT  
0.1

DEMO-BOOL  
✓

DEMO-DICT  
{"a": 0, "b": 1}

DEMO-LIST-OF-FLOATS  
0.1,0.2,0.3

DEMO-IMAGE  
[Image of a sunset over a field]

DEMO-AUDIO  
[Audio player: 0:00 / 0:33]

DEMO-VIDEO  
[Video player: 0:00 / 0:30]

DEMO-STR  
some-string 0.73s

DEMO-IMAGE  
[Image of a sunset over a field]

DEMO-INT-RANGE  
50

DEMO-VIDEO  
[Video player: 0:00 / 0:30]

DEMO-AUDIO  
[Audio player: 0:00 / 0:33]

DEMO-FLOAT  
0.1

DEMO-DICT  
root: {} 2 keys  
a: 0  
b: 1

DEMO-BOOL  
True

DEMO-STR-CHOICE  
choice2

DEMO-INT  
1

DEMO-LIST-OF-FLOATS  
[0.1, 0.2, 0.3]

CLASSIFICATION SCORES

class3

class3	35%
class0	20%
class2	18%
class1	14%
class4	14%

Screenshot Flag

# Roadmap

- Allow API to be restricted to just inferences
  - i.e. `deepaas-run -predict-only`
- Implement compatibility support with KServe API
- Improve integration with user models
  - Currently integrated via Python entry points, support also `@decorators`
  - Remove explicit field definition, use Python type hints
- Improve task off loading
  - Drop (complex, custom) code to support process pools
  - Use Dask (alpha version) or Ray (exploring), with the possibility to offload task to existing clusters

# Roadmap

- Implement self-contained basic security settings
  - i.e. not rely only on flat, provide token-based basic authentication
- DEEPaaS UI
  - Allow to serve several models from same endpoint
- Integration with ML provenance systems
  - Exploring optional MLFlow integration
- Implementation of automated tooling to create a Docker container + API + model

# Wrap up

- Do you have a suggestion? New functionality?
  - <https://deepaas.readthedocs.io>
- DEEPaaS API repository and documentation
  - <https://github.com/indigo-dc/DEEPaaS>
  - <https://deepaas.readthedocs.io>
- DEEPaaS UI repository
  - [https://github.com/deephdc/deepaas\\_ui](https://github.com/deephdc/deepaas_ui)
- FLAAT (for authentication)
  - <https://github.com/indigo-dc/flaat>
- Integration example
  - [https://github.com/deephdc/demo\\_app](https://github.com/deephdc/demo_app)



AI4

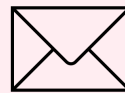
eOSC



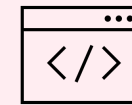
Co-funded by  
the European Union



AI4EOSC



ai4eosc-po@listas.csic.es



ai4eosc.eu

# Reach us!

Thank you for your attention

Project Coordinator: Álvaro López García - [aloga@ifca.unican.es](mailto:aloga@ifca.unican.es)