Using the DEEP-Hybrid-Datacloud platform

Remote webinar for EGI-ACE
05 April 2022

Ignacio Heredia
iheredia@ifca.unican.es
Instituto de Física de Cantabria
(CSIC-UC)

Introduction - The project

- The project was carried out with European Horizon 2020 funds.
- The project provides **new generation of e-infrastructures** that harness latest generation technologies, supporting deep learning and other intensive computing techniques to exploit very large data sources.
- It aims to **lower the adoption barriers** for new communities and users, satisfying the needs of both research, education communities and citizen science.

Project partners:

CSIC
CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

LIP

INFN
Istituto Nazionale di Fisica Nucleare

PSNC

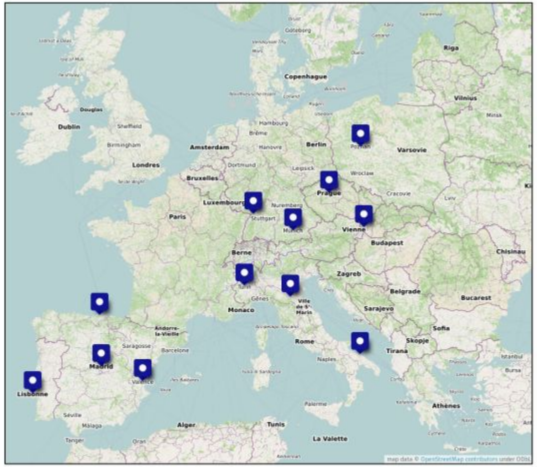
KIT
Karlsruhe Institute of Technology

cesnet

ÚSTAV INFORMATIKY
SLOVENSKÁ AKADÉMIA VIED

AtoS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA



Introduction - The users

Basic

No machine learning knowledge. Just give me a working model to make predictions.

We offer:

- ➔ a **catalogue** full of ready-to-use modules to perform inference with your data
- ➔ an **API** to easily interact with the services
- ➔ solutions to run the inference in local or **Cloud resources**
- ➔ the ability to develop complex topologies by **composing different modules**

Intermediate

I want to retrain a working model on my personal dataset.

We offer:

- ➔ the ability to train out-of-the-box a module of the **catalogue** on your personal dataset
- ➔ an **API** to easily interact with the model
- ➔ **data storage** resources to access your dataset (DEEP-Nextcloud, OneData, …)
- ➔ the ability to deploy the developed service on **Cloud resources**
- ➔ the ability to **share the service** with other users in the user's catalogue

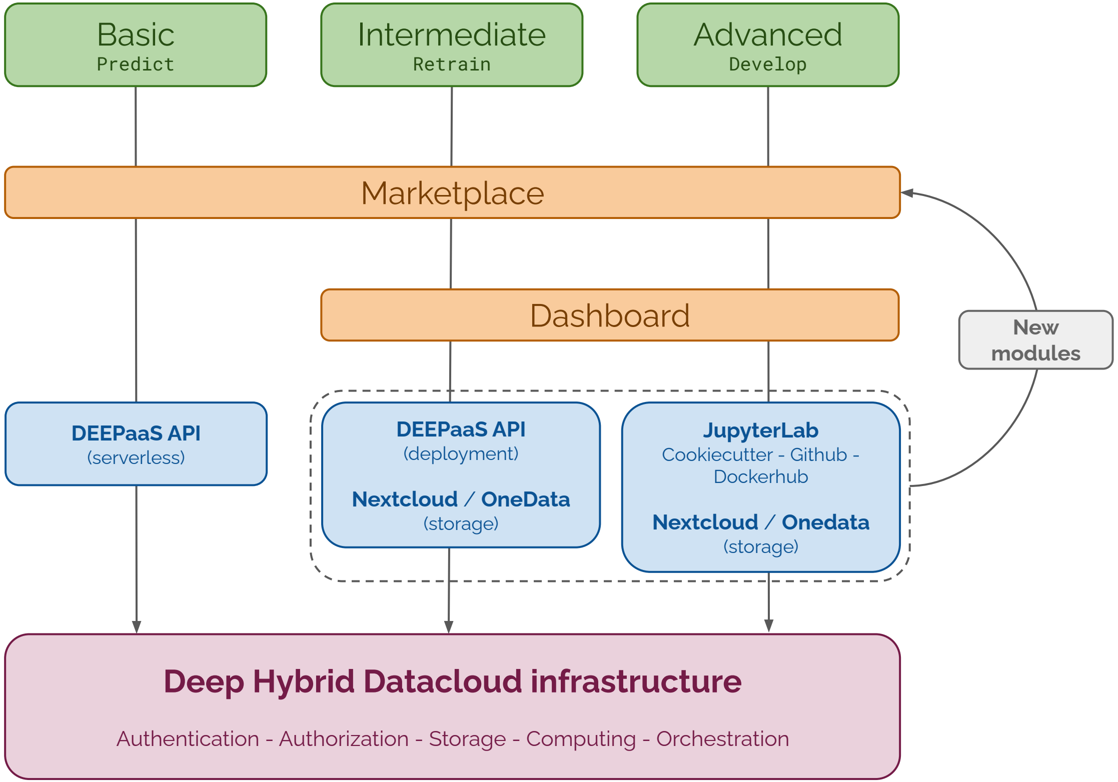
Advanced

I want to develop my custom Deep Learning model.

We offer:

- ➔ a ready-to-use environment with the **main DL frameworks** running in a dockerized solution running on different types of hardware (CPUs, GPUs, etc)
- ➔ **data storage** resources to access your dataset (DEEP-Nextcloud, OneData, …)
- ➔ the ability to deploy the developed module on **Cloud resources**
- ➔ the ability to share the module with other users in the open **catalogue**
- ➔ the possibility to integrate your module with the **API** to enable easier user interaction

Introduction - The users



Basic
Predict

Intermediate
Retrain

Advanced
Develop

Marketplace

Dashboard

New modules

DEEPaaS API
(serverless)

DEEPaaS API
(deployment)

Nextcloud / OneData
(storage)

JupyterLab
Cookiecutter - Github - Dockerhub

Nextcloud / Onedata
(storage)

Deep Hybrid Datacloud infrastructure

Authentication - Authorization - Storage - Computing - Orchestration

Introduction - Useful links

Homepage	https://deep-hybrid-datacloud.eu/	
Marketplace	https://marketplace.deep-hybrid-datacloud.eu/	
Dashboard	https://train.deep-hybrid-datacloud.eu/	
Github	https://github.com/deephdc	
DockerHub	https://hub.docker.com/u/deephdc/	
Documentation	https://docs.deep-hybrid-datacloud.eu/en/latest/	(* these slides are available here)
NextCloud	https://nc.deep-hybrid-datacloud.eu/	

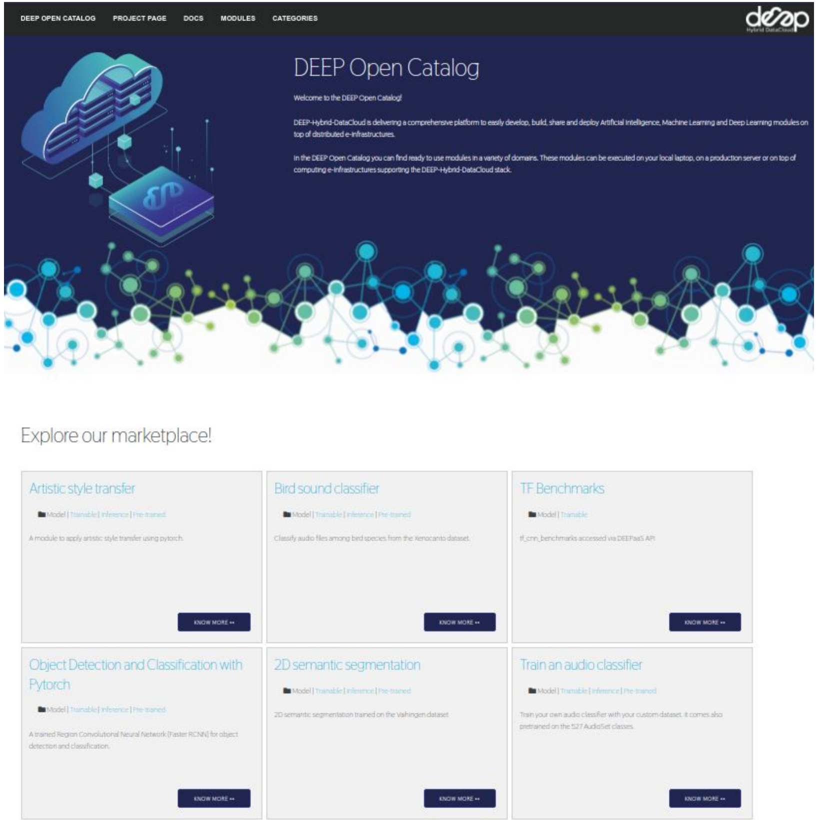
Introduction - Webinar outline

1. **Exploring the Marketplace**
2. **Using the Dashboard**
    a. Deploying a module
    b. Making inference
    c. Retraining a module on a new dataset
3. **Developing a new module**
    a. Deploying the DEEP development environment
    b. Using the cookiecutter
    c. Integrating it with DEEPaaS API
    d. Adding the model to the CI pipeline
    e. Adding the model to the Marketplace
4. **What’s next?**
    a. New DEEPaaS features
    b. Friendlier UI for module inference
    c. Training Dashboard

1

Exploring the Marketplace

The Marketplace



DEEP Open Catalog

Welcome to the DEEP Open Catalog!

DEEP-Hybrid-DataCloud is delivering a comprehensive platform to easily develop, build, share and deploy Artificial Intelligence, Machine Learning and Deep Learning modules on top of distributed e-Infrastructures.

In the DEEP Open Catalog you can find ready to use modules in a variety of domains. These modules can be executed on your local laptop, on a production server or on top of computing e-Infrastructures supporting the DEEP-Hybrid-DataCloud stack.

Explore our marketplace!

Artistic style transfer

Model | Trainable | Inference | Pre-trained

A module to apply artistic style transfer using pytorch.

Bird sound classifier

Model | Trainable | Inference | Pre-trained

Classify audio files among bird species from the Xenocanto dataset.

TF Benchmarks

Model | Trainable

tf_cnn_benchmarks accessed via DEEPaaS API

Object Detection and Classification with Pytorch

Model | Trainable | Inference | Pre-trained

A trained Region Convolutional Neural Network (Faster RCNN) for object detection and classification.

2D semantic segmentation

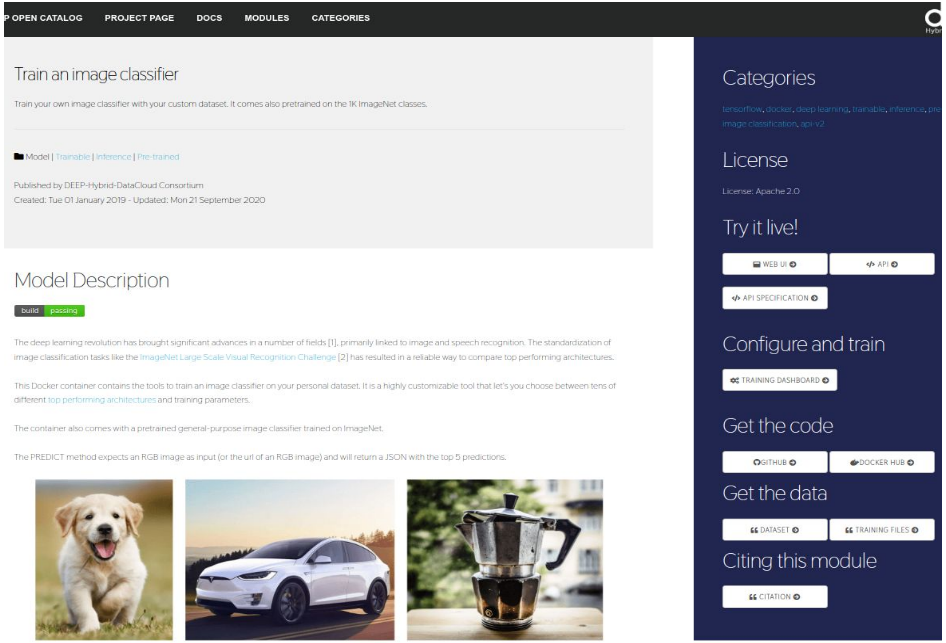
Model | Trainable | Inference | Pre-trained

2D semantic segmentation trained on the Vaihingen dataset

Train an audio classifier

Model | Trainable | Inference | Pre-trained

Train your own audio classifier with your custom dataset. It comes also pretrained on the 527 AudioSet classes.



Train an image classifier

Train your own image classifier with your custom dataset. It comes also pretrained on the 1K ImageNet classes.

Model | Trainable | Inference | Pre-trained

Published by DEEP-Hybrid-DataCloud Consortium
Created: Tue 01 January 2019 - Updated: Mon 21 September 2020

Model Description

The deep learning revolution has brought significant advances in a number of fields [1], primarily linked to image and speech recognition. The standardization of image classification tasks like the ImageNet Large Scale Visual Recognition Challenge [2] has resulted in a reliable way to compare top performing architectures.

This Docker container contains the tools to train an image classifier on your personal dataset. It is a highly customizable tool that let's you choose between tens of different top performing architectures and training parameters.

The container also comes with a pretrained general-purpose image classifier trained on ImageNet.

The PREDICT method expects an RGB image as input (or the url of an RGB image) and will return a JSON with the top 5 predictions.

Categories

tensorflow, docker, deep learning, trainable, inference, pre
image classification, api-v2

License

License: Apache 2.0

Try it live!

Configure and train

Get the code

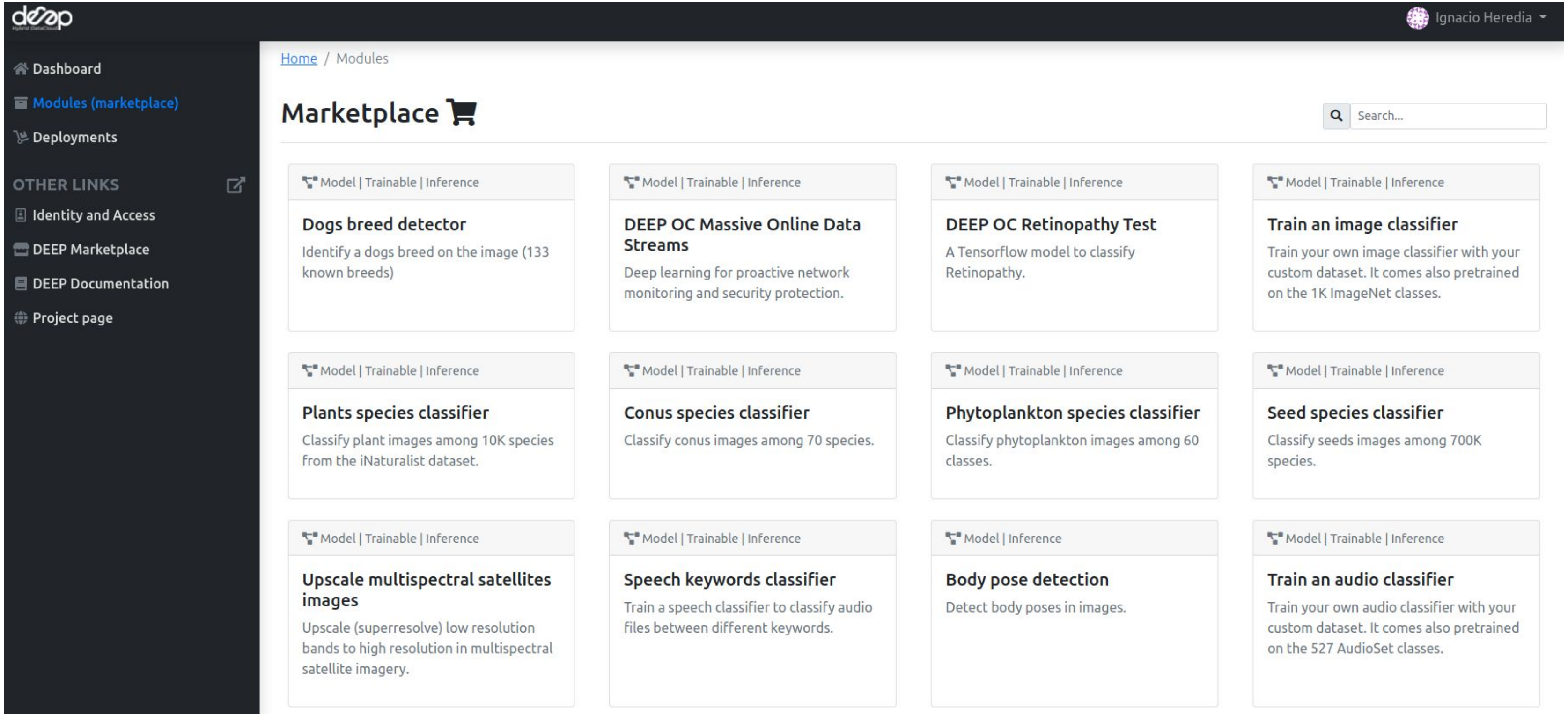
Get the data

Citing this module

2

The Dashboard

The Dashboard - Module Overview



Home / Modules

Marketplace

Model | Trainable | Inference

Dogs breed detector

Identify a dogs breed on the image (133 known breeds)

Model | Trainable | Inference

DEEP OC Massive Online Data Streams

Deep learning for proactive network monitoring and security protection.

Model | Trainable | Inference

DEEP OC Retinopathy Test

A Tensorflow model to classify Retinopathy.

Model | Trainable | Inference

Train an image classifier

Train your own image classifier with your custom dataset. It comes also pretrained on the 1K ImageNet classes.

Model | Trainable | Inference

Plants species classifier

Classify plant images among 10K species from the iNaturalist dataset.

Model | Trainable | Inference

Conus species classifier

Classify conus images among 70 species.

Model | Trainable | Inference

Phytoplankton species classifier

Classify phytoplankton images among 60 classes.

Model | Trainable | Inference

Seed species classifier

Classify seeds images among 700K species.

Model | Trainable | Inference

Upscale multispectral satellites images

Upscale (superresolve) low resolution bands to high resolution in multispectral satellite imagery.

Model | Trainable | Inference

Speech keywords classifier

Train a speech classifier to classify audio files between different keywords.

Model | Inference

Body pose detection

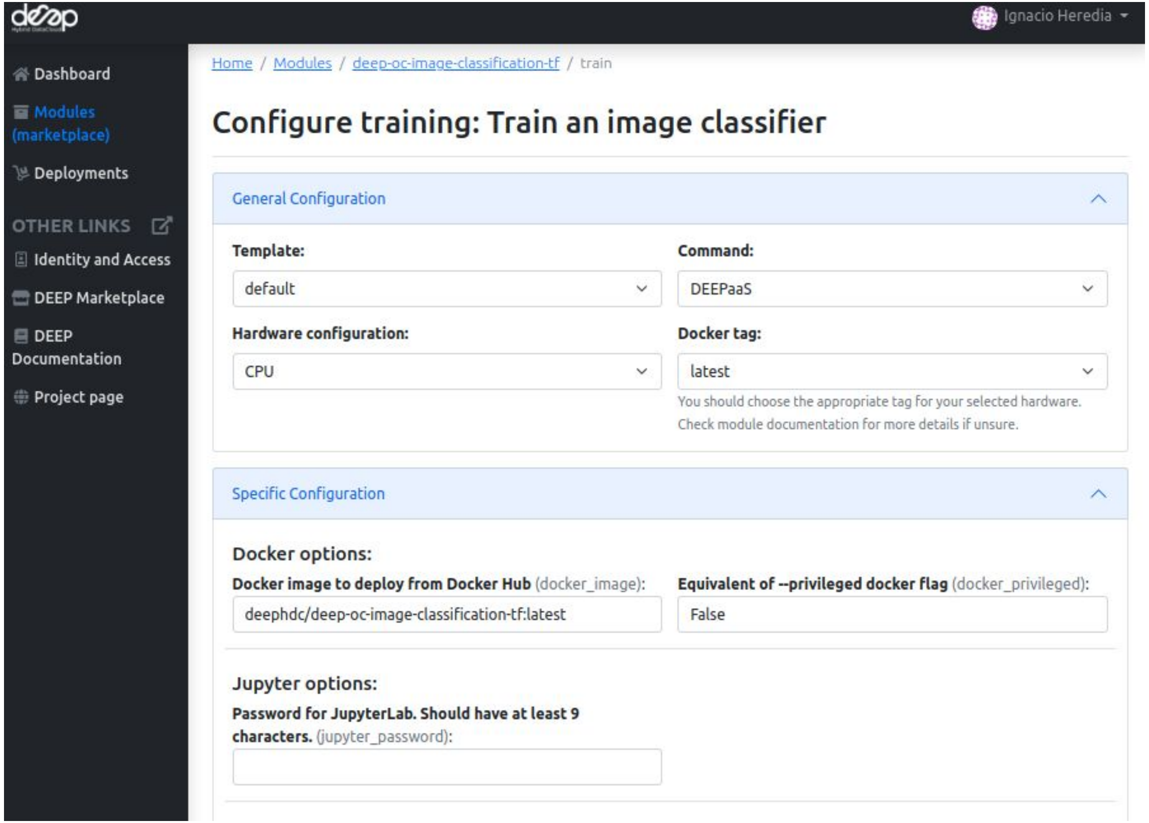
Detect body poses in images.

Model | Trainable | Inference

Train an audio classifier

Train your own audio classifier with your custom dataset. It comes also pretrained on the 527 AudioSet classes.

The Dashboard - Deploying a module



Configure training: Train an image classifier

Template: default

Command: DEEPaaS

Hardware configuration: CPU

Docker tag: latest

You should choose the appropriate tag for your selected hardware. Check module documentation for more details if unsure.

Docker options:

Docker image to deploy from Docker Hub (docker_image): deephdc/deep-oc-image-classification-tf:latest

Equivalent of --privileged docker flag (docker_privileged): False

Jupyter options:

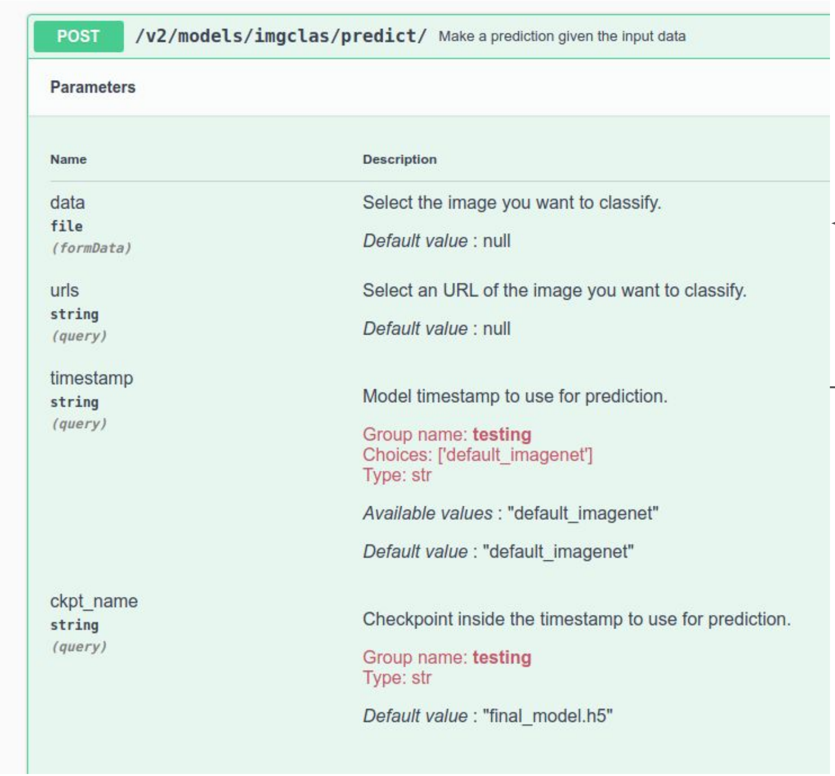
Password for JupyterLab. Should have at least 9 characters. (jupyter_password):

Configurable options

- **docker image** (from deep-oc, but also custom docker images)
- **hardware** (#cpus, #gpus, RAM)
- **storage** (OneData, Nextcloud volumes)
- **services** (DEEPaaS, JupyterLab)

The Dashboard - Making inference

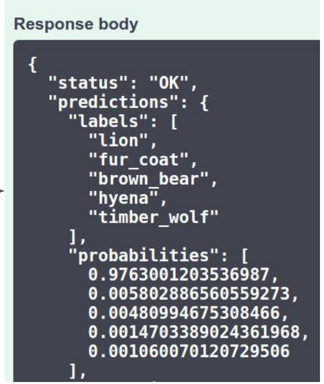
Launch `image-classification-tf` module with DEEPaaS.



POST /v2/models/imgclas/predict/ Make a prediction given the input data

Parameters

Name	Description
data `file` *(formData)*	Select the image you want to classify. *Default value* : null
urls `string` *(query)*	Select an URL of the image you want to classify. *Default value* : null
timestamp `string` *(query)*	Model timestamp to use for prediction. Group name: **testing** Choices: ['default_imagenet'] Type: str *Available values* : "default_imagenet" *Default value* : "default_imagenet"
ckpt_name `string` *(query)*	Checkpoint inside the timestamp to use for prediction. Group name: **testing** Type: str *Default value* : "final_model.h5"



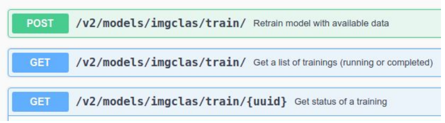
Response body

```
{
  "status": "OK",
  "predictions": {
    "labels": [
      "lion",
      "fur_coat",
      "brown_bear",
      "hyena",
      "timber_wolf"
    ],
    "probabilities": [
      0.9763001203536987,
      0.005802886560559273,
      0.004809946753084466,
      0.0014703389024361968,
      0.001060070120729506
    ],
```

The Dashboard - Retraining a module

1) Launch `image-classification-tf` module with JupyterLab (remember adding password).
2) Copy some demo files to make a mock dataset.
3) Terminal: `deepaas-run --listen-ip 0.0.0.0` to launch DEEPaaS.

DEEPaaS



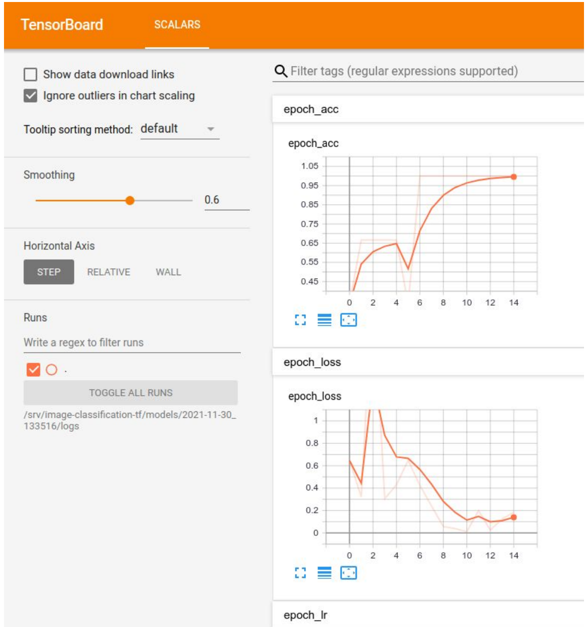
POST /v2/models/imgclas/train/ Retrain model with available data

GET /v2/models/imgclas/train/ Get a list of trainings (running or completed)

GET /v2/models/imgclas/train/{uuid} Get status of a training

Monitor

TensorBoard



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

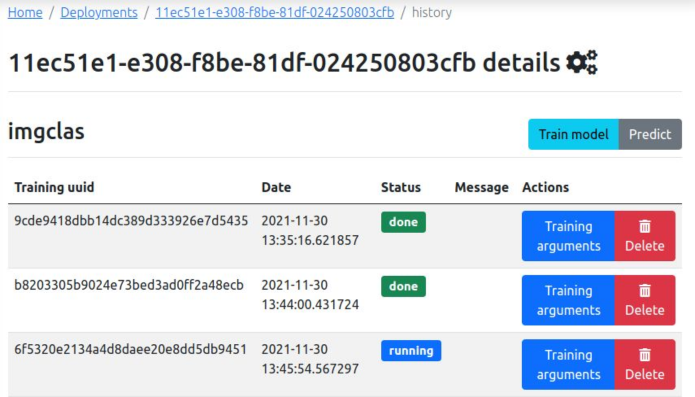
☑ ○ .

TOGGLE ALL RUNS

/srv/image-classification-tf/models/2021-11-30_133516/logs

Filter tags (regular expressions supported)

Training history



Home / Deployments / 11ec51e1-e308-f8be-81df-024250803cfb / history

11ec51e1-e308-f8be-81df-024250803cfb details

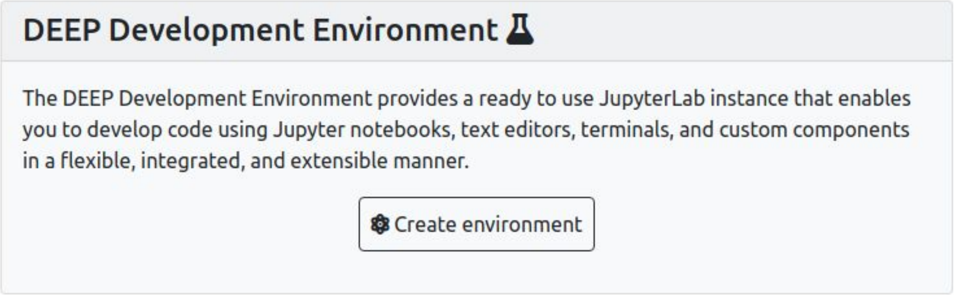
imgclas

Training uuid	Date	Status	Message	Actions
9cde9418dbb14dc389d333926e7d5435	2021-11-30 13:35:16.621857	done		Training arguments Delete
b8203305b9024e73bed3ad0ff2a48ecb	2021-11-30 13:44:00.431724	done		Training arguments Delete
6f5320e2134a4d8daee20e8dd5db9451	2021-11-30 13:45:54.567297	running		Training arguments Delete

3

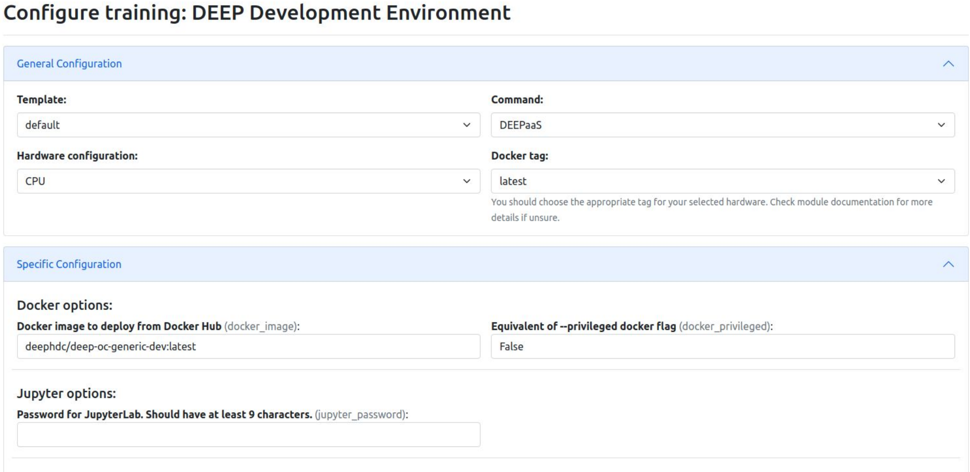
Develop your module

Developing - DEEP Development Environment



DEEP Development Environment

The DEEP Development Environment provides a ready to use JupyterLab instance that enables you to develop code using Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner.



Configure training: DEEP Development Environment

General Configuration

Template: default

Command: DEEPaaS

Hardware configuration: CPU

Docker tag: latest

You should choose the appropriate tag for your selected hardware. Check module documentation for more details if unsure.

Specific Configuration

Docker options:

Docker image to deploy from Docker Hub (docker_image): deephdc/deep-oc-generic-dev:latest

Equivalent of --privileged docker flag (docker_privileged): False

Jupyter options:

Password for JupyterLab. Should have at least 9 characters. (jupyter_password):

Configurable options

- **docker image** (from deep-oc, but also custom docker images). Eg:
  - **Tensorflow** docker
  - **Pytorch** docker
  - ...
- **hardware** (#cpus, #gpus, RAM)
- **storage** (OneData, Nextcloud volumes)
- **services** (DEEPaaS, **JupyterLab**)

Developing - DEEP Cookiecutter

This is the easiest way to develop any new module from scratch as it will take care of generating all the nitty-gritty details that we will cover in the following slides (entrypoints, files, Jenkinsfile, Dockerfile, etc).

- Use the command: `cookiecutter https://github.com/indigo-dc/cookiecutter-data-science`
- Answer questions:
  - Project name, description, version, license type
  - Author name, email, Github account
  - Dockerhub account, Docker base image
- This will generate two folders. Eg:
  - `mymodule`: This is where the project code is located

    → Example: https://github.com/deephdc/image-classification-tf
  - `DEEP-OC-mymodule`: This contains the Dockerfile of the project

    → Example: https://github.com/deephdc/DEEP-OC-image-classification-tf

Developing - Integrating with DEEPaaS

- Head over to `mymodule`. Any module that wants to integrate with DEEPaaS should have two minimum requirements:

  - it should define a file (eg. `mymodule/mymodule/api.py`) with the functions to interact with the module. These functions should define:

■ the model metadata	`get_metadata()`
■ the input args for training	`get_train_args()`
■ the input args for prediction	`get_predict_args()`
■ the response structure for prediction	`schema`
■ the train function	`train()`
■ the predict function	`predict ()`
■ a model warming function for prediction	`warm()`

    → Minimal example: https://github.com/deephdc/demo_app/blob/master/demo_app/api.py

    → Full example: https://github.com/deephdc/image-classification-tf/blob/master/imgclas/api.py

  - it should define an entrypoint in `mymodule/setup.cfg` pointing to that file

    → Example: https://github.com/deephdc/demo_app/blob/master/setup.cfg#L25-L27

Developing - Customizing the Dockerfile

- Head over to `DEEP-OC-mymodule` and modify the Dockerfile following your needs:
  - install additional packages,
  - change the base image,
  - etc.

Developing - Continuous Integration

- Both `mymodule` and `DEEP-OC-mymodule` have their respective `Jenkinsfile` that define the actions to be taken when a change is committed to the repos.
- Typical workflows:
  - `mymodule/Jenkinsfile` will:
    - run PEP8 style analysis
    - trigger of `DEEP-OC-mymodule/Jenkinsfile`.

    → Example: https://github.com/deephdc/image-classification-tf/blob/master/Jenkinsfile

  - `DEEP-OC-mymodule/Jenkinsfile` will:
    - build Docker images for different branches (train/test) and different hardware (cpu/gpu)
    - upload the image to DockerHub
    - build Docker images of other dependent modules. For example, changes in the code of image-classification should rebuild all Docker images of applications that were trained with that code (plant classifier, seed classifier, etc).
    - refresh the module page in the Marketplace (see next step)

    → Example: https://github.com/deephdc/DEEP-OC-image-classification-tf/blob/master/Jenkinsfile

Developing - Integrating to the Marketplace

- Head over to `DEEP-OC-mymodule` and modify `metadata.json` with the info relevant to your module. This is the information that will appear in the Marketplace page.
- Make a Pull Request to add your module here. This will create the Jenkins pipeline for your module and will add the module to the Marketplace and the Training Dashboard.

🎉 **Congratulations, you're done!** 🎉

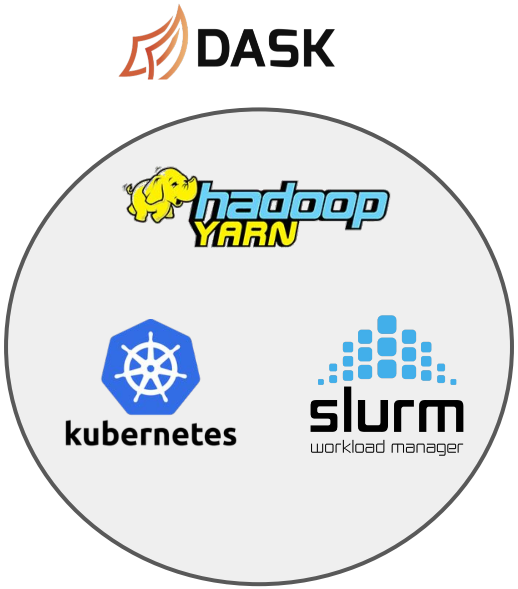
4

What’s next?

What's next? - New DEEPaaS features

- Integration with Dask

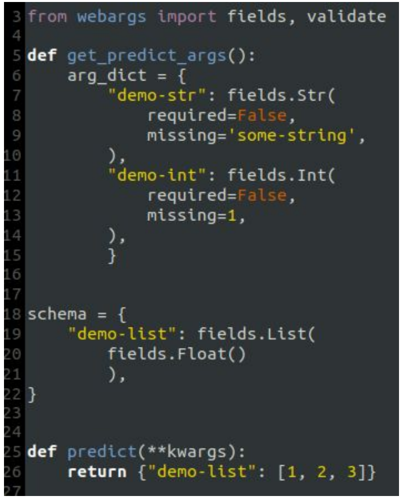
Mature



- Easier module integration via decorators/hints

Midterm

Before (webargs)



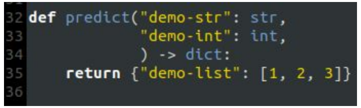
```
from webargs import fields, validate

def get_predict_args():
    arg_dict = {
        "demo-str": fields.Str(
            required=False,
            missing='some-string',
        ),
        "demo-int": fields.Int(
            required=False,
            missing=1,
        ),
        }


schema = {
    "demo-list": fields.List(
        fields.Float()
        ),
}


def predict(**kwargs):
    return {"demo-list": [1, 2, 3]}
```

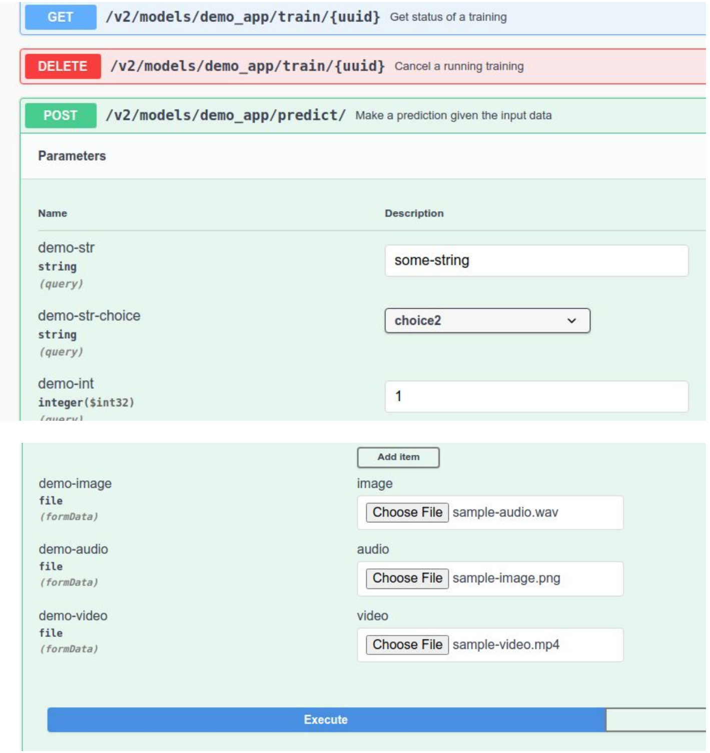
After (type hints)



```
def predict("demo-str": str,
            "demo-int": int,
            ) -> dict:
    return {"demo-list": [1, 2, 3]}
```

What's next? - Friendlier inference UI

Before (Swagger UI)



GET /v2/models/demo_app/train/{uuid} Get status of a training

DELETE /v2/models/demo_app/train/{uuid} Cancel a running training

POST /v2/models/demo_app/predict/ Make a prediction given the input data

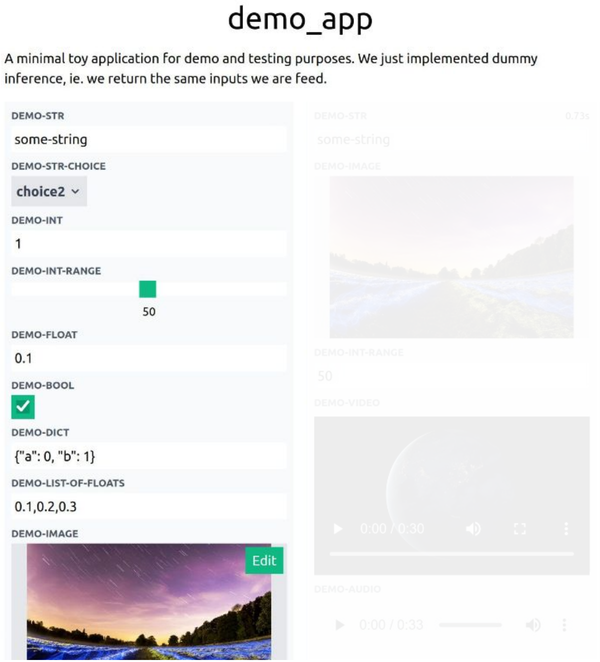
Parameters

Name	Description
demo-str string *(query)*	some-string
demo-str-choice string *(query)*	choice2
demo-int integer($int32) *(query)*	1
	Add item
demo-image file *(formData)*	image Choose File sample-audio.wav
demo-audio file *(formData)*	audio Choose File sample-image.png
demo-video file *(formData)*	video Choose File sample-video.mp4

Execute

Inputs

After (Gradio based) Mature



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
Edit

DEMO-STR 0.73s
some-string

DEMO-IMAGE

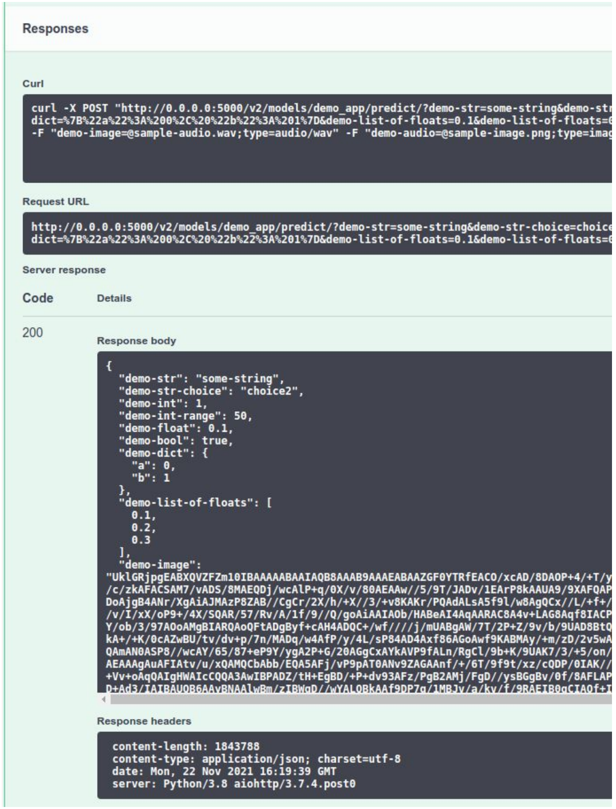
DEMO-INT-RANGE
50

DEMO-VIDEO
0:00 / 0:30

DEMO-AUDIO
0:00 / 0:33

What's next? - Friendlier inference UI

Before (Swagger UI)



Responses

Curl

```
curl -X POST "http://0.0.0.0:5000/v2/models/demo_app/predict/?demo-str=some-string&demo-str
dict=%7B%22a%22%3A%200%2C%20%22b%22%3A%201%7D&demo-list-of-floats=0.1&demo-list-of-floats=0
-F "demo-image=@sample-audio.wav;type=audio/wav" -F "demo-audio=@sample-image.png;type=imag
```

Request URL

```
http://0.0.0.0:5000/v2/models/demo_app/predict/?demo-str=some-string&demo-str-choice=choice
dict=%7B%22a%22%3A%200%2C%20%22b%22%3A%201%7D&demo-list-of-floats=0.1&demo-list-of-floats=0
```

Server response

Code	Details
200	Response body

```
{
  "demo-str": "some-string",
  "demo-str-choice": "choice2",
  "demo-int": 1,
  "demo-int-range": 50,
  "demo-float": 0.1,
  "demo-bool": true,
  "demo-dict": {
    "a": 0,
    "b": 1
  },
  "demo-list-of-floats": [
    0.1,
    0.2,
    0.3
  ],
  "demo-image":
"UklGRjpgEABXQVZFZm10IBAAAAABAAIAQB8AAAB9AAAEABAAZGF0YTRfEACO/xcAD/8DAOP+4/+T/y
/c/zkAFACSAM7/vADS/8MAEQDj/wcAlP+q/0X/v/80AEAAw//5/9T/JADv/1EArP8kAAUA9/9XAFQAP
DoAjgB4ANr/XgAiAJMAzP8ZAB//CgCr/2X/h/+X//3/+v8KAKr/PQAdALsA5f9l/w8AgQCx//L/+f+/
/v/I/xX/oP9+/4X/SQAR/57/Rv/A/1f/9//Q/goAiAAIAOb/HABeAI4AqAARAC8A4v+LAG8Aqf8IACP
Y/ob/3/97AOoAMgBIARQAoQFtADgByf+cAH4ADQC+/wf////j/mUABgAW/7T/2P+Z/9v/b/9UAD8BtQ
kA+/+K/0cAZwBU/tv/dv+p/7n/MADq/w4AfP/y/4L/sP84AD4Axf86AGoAwf9KABMAy/+m/zD/2v5wA
QAmAN0ASP8//wcAY/65/87+eP9Y/ygA2P+G/20AGgCxAYkAVP9fALn/RgCl/9b+K/9UAK7/3/+5/on/
AEAAAgAuAFIAtv/u/xQAMQCbAbb/EQA5AFj/vP9pAT0ANv9ZAGAAnf/+/6T/9f9t/xz/cQDP/0IAK//
+Vv+oAqQAIgHWAIcCQQA3AwIBPADZ/tH+EgBD/+P+dv93AFz/PgB2AMj/FgD//ysBGgBv/0f/8AFLAP
D+Ad3/IAIBAUOB6AAvBNAAlwBm/zIBWgD//wYALQBkAAf9DP7q/1MBJv/a/kv/f/9RAEIB0gCIAOf+I
```

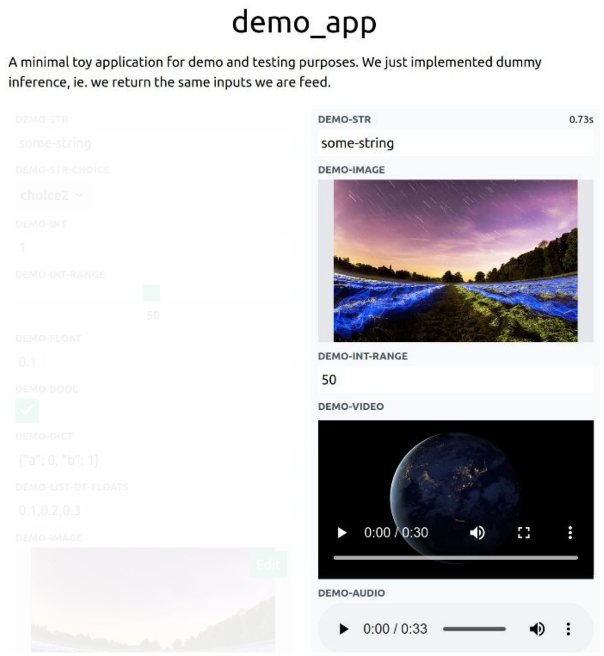
Response headers

```
content-length: 1843788
content-type: application/json; charset=utf-8
date: Mon, 22 Nov 2021 16:19:39 GMT
server: Python/3.8 aiohttp/3.7.4.post0
```

Outputs

After (Gradio based)

Mature



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

DEMO-STR 0.73s
some-string

DEMO-IMAGE

DEMO-INT-RANGE
50

DEMO-VIDEO

DEMO-AUDIO

What's next? - Training dashboard

- Organizing training run in experiments
    - hyperparameter optimization
    - easier side-by-side comparison of training runs

Midterm

- Richer module metadata language, to keep track of:
    - training datasets
    - models
    - training execution pipelines

Project

Questions

Hybrid DataCloud

European Commission

Horizon 2020
European Union funding
for Research & Innovation

CSIC
Spanish Council of Research

DEEP-Hybryd-DataCloud has received funding from the European Horizon 2020 research and innovation programme under grant agreement N°777435