DEEPHEALTH

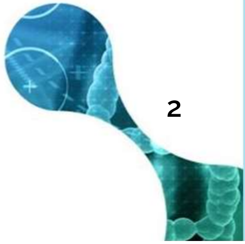
Tensor Manipulation and Data Augmentation with ECVL

Costantino Grana - UNIMORE

Winter School 25/01/2022

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Contents



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

From ECVL to EDDL

- ECVL allows easy integration and exchange of data with EDDL.
- The ECVL - EDDL interfacing is based on two main functions: *ImageToTensor* and *TensorToImage*.
- *ImageToTensor* transforms an ECVL Image in a EDDL Tensor, converting its data to floating point numbers and rearranging its channels to "xy[c/z/o]", which is how EDDL Tensor handles data in color images. Finally, Image data are copied into Tensor data.

```
import pyecvl.ecvl as ecvl
from pyeddl.tensor import Tensor

image = ecvl.ImRead("/mnt/data/winter_school/lena.png")
t = ecvl.ImageToTensor(image)
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

From EDDL to ECVL

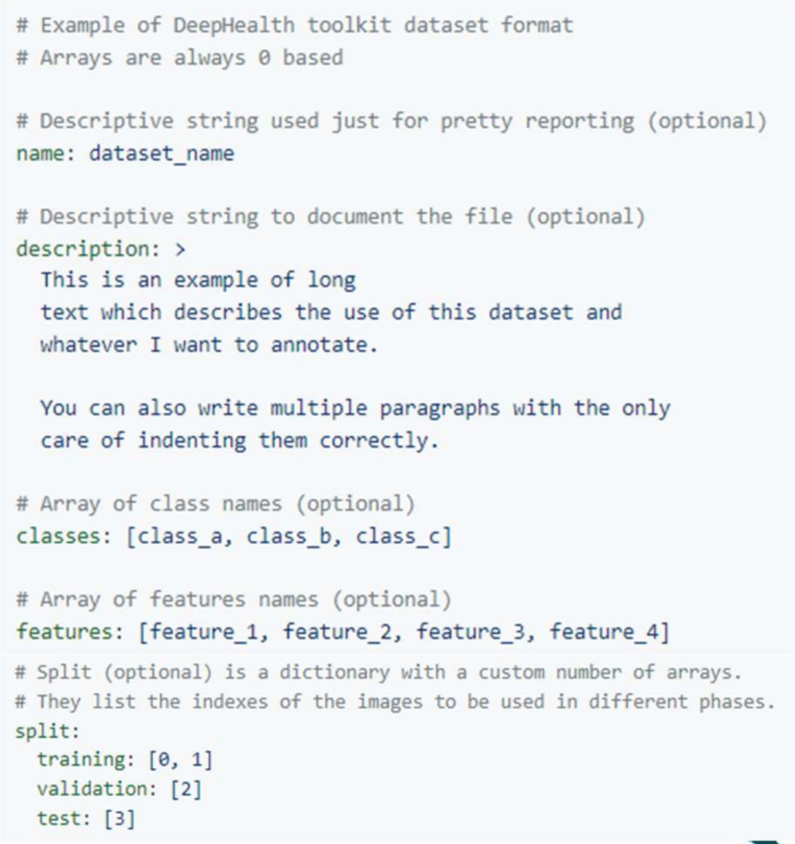
- *TensorToImage* creates a float “xyo” raw Image with *none* color space, and copies Tensor data into the Image data.
- *TensorToView* is also available in order to avoid the copy of the data.

```
import pyecvl.ecvl as ecvl
from pyeddl.tensor import Tensor

image = ecvl.ImRead("/mnt/data/winter_school/lena.png")
t = ecvl.ImageToTensor(image)
image_from_tensor = ecvl.TensorToImage(t)
view_from_tensor = ecvl.TensorToView(t)
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Dataset Format

- A simple and flexible YAML syntax to describe a dataset for the DeepHealth libraries (EDDL/ECVL), in order to provide a unified way for loading data.
- Complete description: [DeepHealth-Toolkit-Dataset-Format]()
- Optional elements: *name*, *description*, *classes* (mandatory in case of a classification task), *features* (additional information related to each image), *split*.



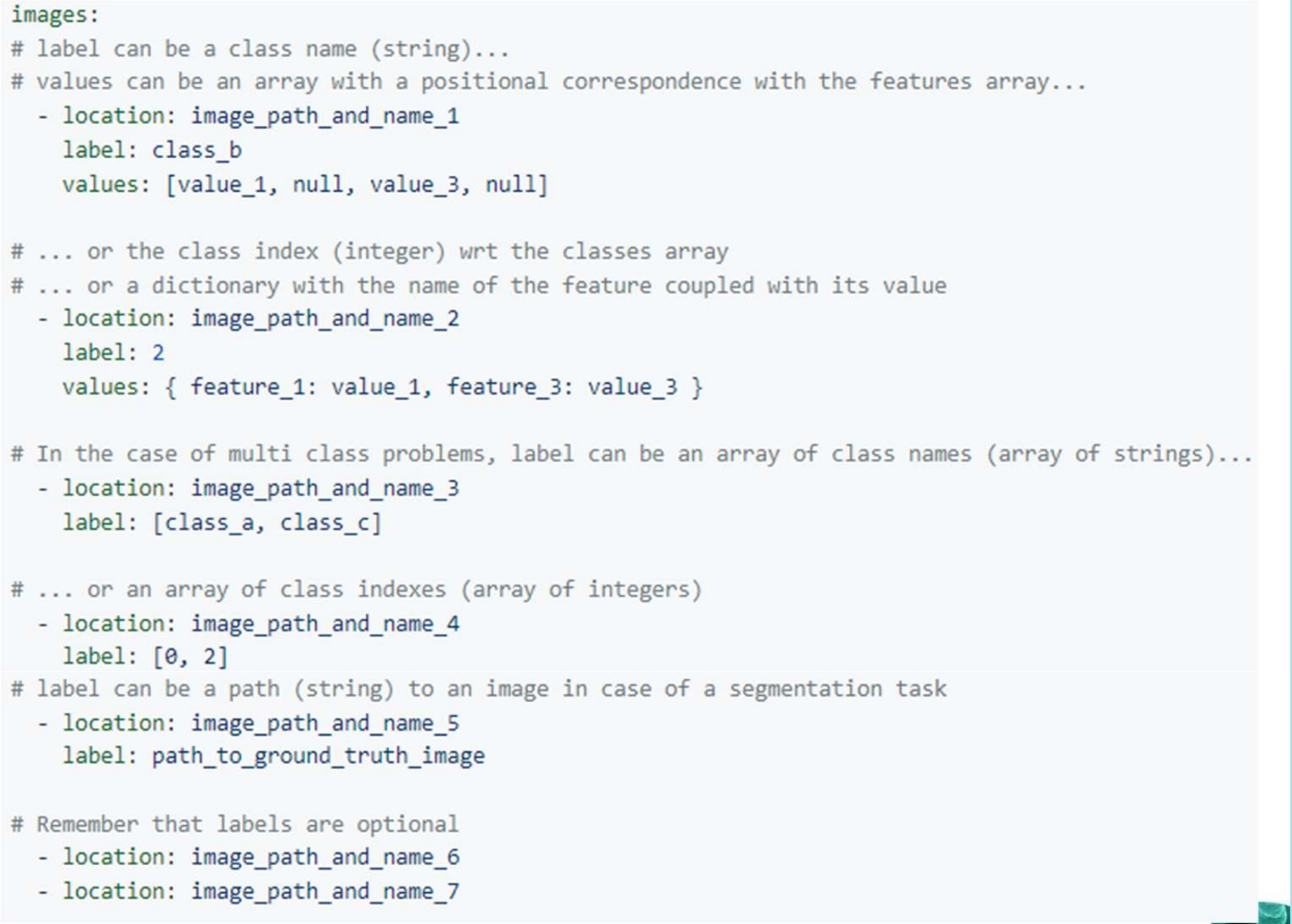
```yaml
# Example of DeepHealth toolkit dataset format
# Arrays are always 0 based

# Descriptive string used just for pretty reporting (optional)
name: dataset_name

# Descriptive string to document the file (optional)
description: >
  This is an example of long
  text which describes the use of this dataset and
  whatever I want to annotate.

  You can also write multiple paragraphs with the only
  care of indenting them correctly.

# Array of class names (optional)
classes: [class_a, class_b, class_c]

# Array of features names (optional)
features: [feature_1, feature_2, feature_3, feature_4]
# Split (optional) is a dictionary with a custom number of arrays.
# They list the indexes of the images to be used in different phases.
split:
  training: [0, 1]
  validation: [2]
  test: [3]
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Dataset Format

DEEPHEALTH

- Mandatory element: the list of the *images* with their *location* and optionally their *label* (a class or a path) and *values*.



```
images:
# label can be a class name (string)...
# values can be an array with a positional correspondence with the features array...
  - location: image_path_and_name_1
    label: class_b
    values: [value_1, null, value_3, null]

# ... or the class index (integer) wrt the classes array
# ... or a dictionary with the name of the feature coupled with its value
  - location: image_path_and_name_2
    label: 2
    values: { feature_1: value_1, feature_3: value_3 }

# In the case of multi class problems, label can be an array of class names (array of strings)...
  - location: image_path_and_name_3
    label: [class_a, class_c]

# ... or an array of class indexes (array of integers)
  - location: image_path_and_name_4
    label: [0, 2]
# label can be a path (string) to an image in case of a segmentation task
  - location: image_path_and_name_5
    label: path_to_ground_truth_image

# Remember that labels are optional
  - location: image_path_and_name_6
  - location: image_path_and_name_7
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Dataset

- ECVL also provides a *Generator* which creates a Dataset file from a directory tree (Dataset-Generator)

```
gcd = ecvl.GenerateClassificationDataset(input_directory)
cls_d = gcd.GetDataset()
cls_d.Dump("classification_dataset.yml")
```

- ECVL allows to parse the Dataset YAML file to create a Dataset object.

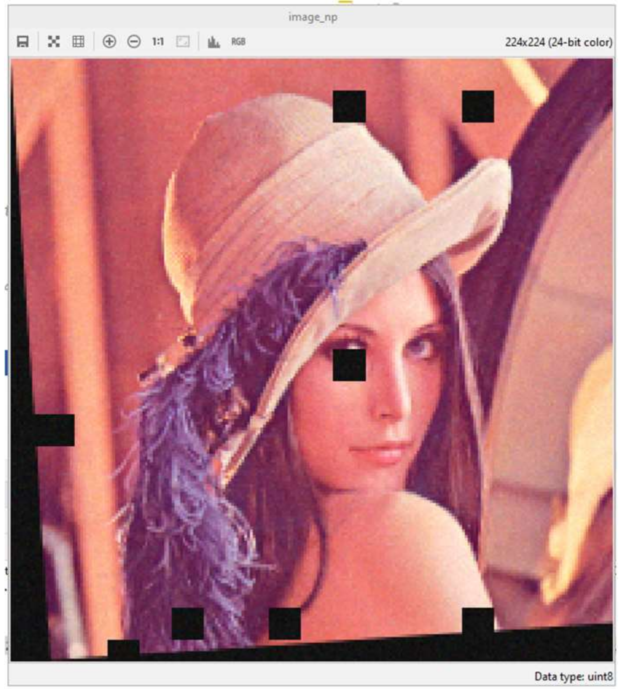
```
input_dataset = "/mnt/data/winter_school/mnist.yml"
d = ecvl.Dataset(input_dataset)
print("name:", d.name_)
print("classes:", d.classes_)
print("features:", d.features_)
print("n. samples:", len(d.samples_))
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Augmentation

- Data Augmentations can be loaded from text…

```
import pyecvl.ecvl as ecvl

image = ecvl.ImRead("/mnt/data/winter_school/lena.png")

AUG_TXT = '''\
SequentialAugmentationContainer
    AugResizeDim dims=(224,224) interp="linear"
    AugRotate angle=[-5,5] center=(0,0) interp="linear"
    AugAdditiveLaplaceNoise std_dev=[0,0.51]
    AugCoarseDropout p=[0,0.05] drop_size=[0.02,0.1] per_channel=0
    AugAdditivePoissonNoise lambda=[0,40]
end
'''

augs_from_text = ecvl.AugmentationFactory.create(AUG_TXT)
augs_from_text.Apply(image)
```

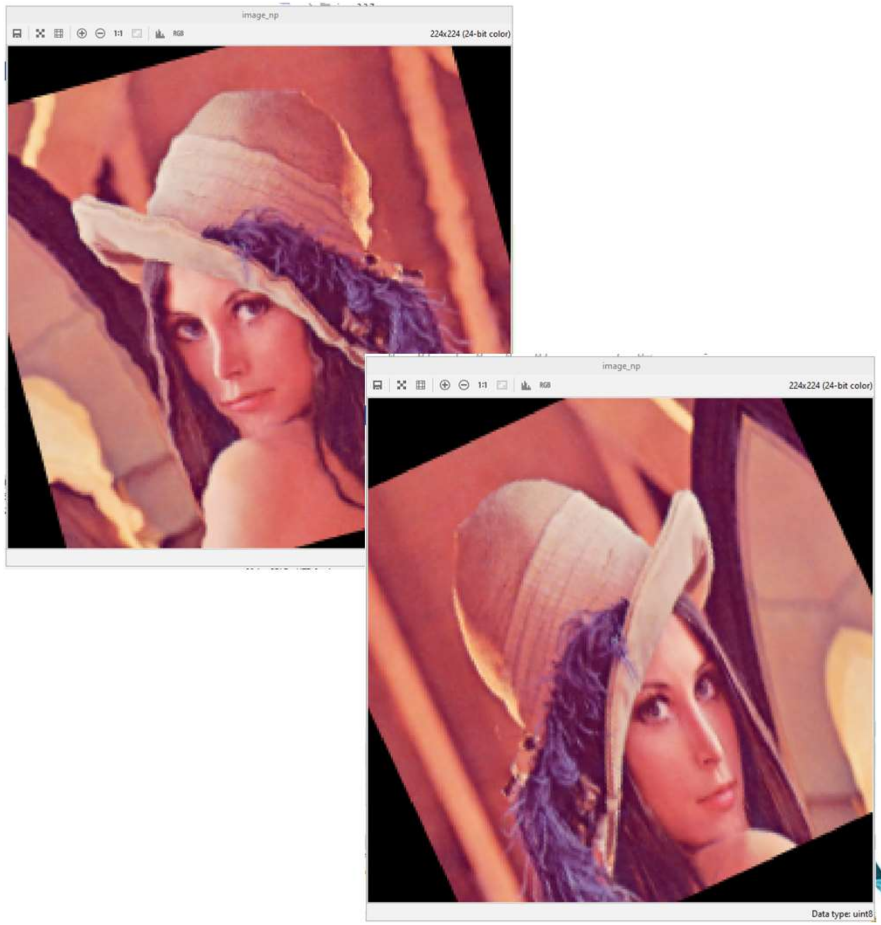


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Augmentation

- …or directly written in the code

```
import pyecvl.ecvl as ecvl

image = ecvl.ImRead("/mnt/data/winter_school/lena.png")
disp_image = ecvl.Image.empty()

training_augs = ecvl.SequentialAugmentationContainer([
    ecvl.AugResizeDim([224, 224]),
    ecvl.AugMirror(.5),
    ecvl.OneOfAugmentationContainer(
        0.3,
        [ecvl.AugElasticTransform([10, 20], [2, 4]),
        ecvl.AugGridDistortion([2, 5], [-0.3, 0.3]),
        ecvl.AugOpticalDistortion([-0.3, 0.3], [-0.1, 0.1])]
    ),
    ecvl.AugRotate([-30, 30]),
])

training_augs.Apply(image)
```



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Augmentation

- The list of all the augmentations is available at https://deephealthproject.github.io/ecvl/master/classecvl_1_1_augmentation.html
- Augmentations must be inserted in a *Container*, which can be
  - *Sequential* → all augmentations in the container will be performed
  - *OneOf* → only one of the augmentations in the container will be performed, with a given probability
- Containers can be nested one inside the other

```
training_augs = ecvl.SequentialAugmentationContainer([
    ecvl.AugResizeDim([224, 224]),
    ecvl.AugMirror(.5),
    ecvl.OneOfAugmentationContainer(
        0.3,
        [ecvl.AugElasticTransform([10, 20], [2, 4]),
        ecvl.AugGridDistortion([2, 5], [-0.3, 0.3]),
        ecvl.AugOpticalDistortion([-0.3, 0.3], [-0.1, 0.1])]
    ),
    ecvl.AugRotate([-30, 30]),
])
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Augmentation

- Different augmentations can be applied to each split defined in the Dataset

```
training_augs = ecvl.SequentialAugmentationContainer([
    ecvl.AugResizeDim([224, 224]),
    ecvl.AugMirror(.5),
    ecvl.OneOfAugmentationContainer(
        0.3,
        [ecvl.AugElasticTransform([10, 20], [2, 4]),
        ecvl.AugGridDistortion([2, 5], [-0.3, 0.3]),
        ecvl.AugOpticalDistortion([-0.3, 0.3], [-0.1, 0.1])]
    ),
    ecvl.AugRotate([-30, 30]),
])

validation_augs = ecvl.SequentialAugmentationContainer([
    ecvl.AugResizeDim([224, 224]),
])

test_augs = ecvl.SequentialAugmentationContainer([
    ecvl.AugResizeDim([224, 224]),
])

ds_augs = ecvl.DatasetAugmentations([training_augs, validation_augs, test_augs])
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Deep Learning Dataset

- The Dataset object is extended by DLDataset, which contains information for the deep learning functionalities, specifically for generating data to feed the neural network

```
input_dataset = "/mnt/data/winter_school/mnist.yml"
batch_size = 64
dataset_augmentations = ecvl.DatasetAugmentations([training_augs, validation_augs, test_augs])
d = ecvl.DLDataset(input_dataset, batch_size, dataset_augmentations, ctype=ecvl.ColorType.GRAY,
                  num_workers=6, queue_ratio_size=5, drop_last=[True, False, False])
```

- In order to create samples in a non-blocking scenario, *num_workers* threads work in parallel to:
  - loading the images
  - apply augmentations
  - convert them to Tensors
  - push the Tensors to a queue from which the main thread can pull a batch for the network

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

ProduceImageLabel

- These steps are performed by the function *ProduceImageLabel,* which in C++ can be overwritten to customize the creation of the batches. For example, this dataset consists of 3D volumes that we want to push into the queue slice by slice

```
class CustomDataset : public DLDataset
{
…
void ProduceImageLabel(DatasetAugmentations& augs, Sample& elem) override
{
    Tensor* label_tensor = nullptr, * image_tensor = nullptr;
    Image img = elem.LoadImage(ctype_, false);
    Image gt = elem.LoadImage(ctype_gt_, true);
    const int slices = img.Channels();

    // Apply chain of augmentations to sample image and corresponding ground truth
    augs.Apply(current_split_, img, gt);

    // Push the slice and its ground truth to the queue
    for (int cur_slice = 0; cur_slice < slices; ++cur_slice) {
        View<DataType::int16> v_volume(img, { 0, 0, cur_slice}, { img.Width(), img.Height(), n_channels_ });
        View<DataType::int16> v_gt(gt, { 0, 0, cur_slice}, { gt.Width(), gt.Height(), n_channels_ });
        ImageToTensor(v_volume, image_tensor);
        ImageToTensor(v_gt, label_tensor);

        queue_.Push(elem, image_tensor, label_tensor);
    }
}
…
}
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Generate Batches

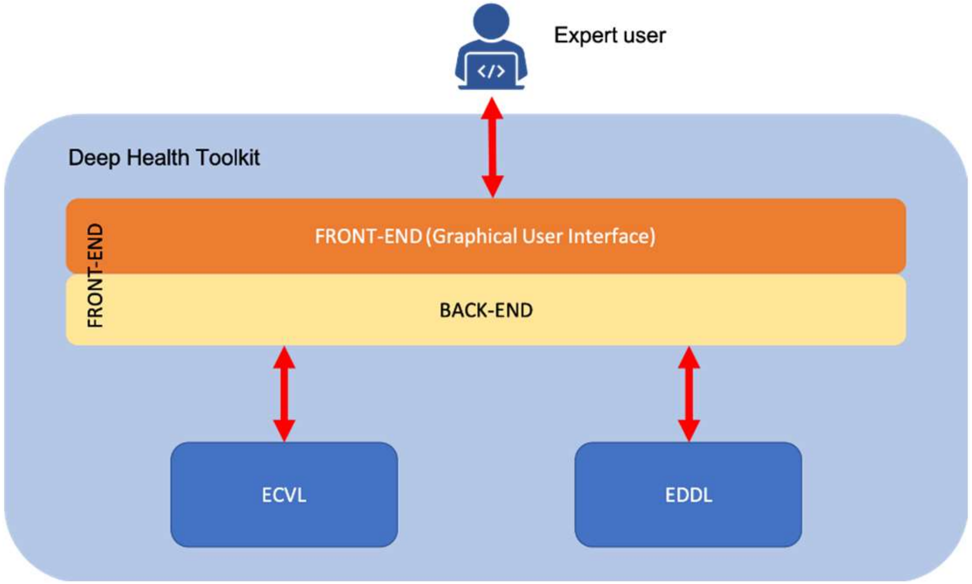
- With the *Start* method the threads are spawned and begin to fill the queue, until the *Stop* method is called.
- The *GetBatch* function will retrieve a vector of samples, which contains information like the name of the files in the batch, the Tensor containing the batch of the prepared images and the Tensor containing the corresponding labels.

```
d.Start()
for b in range(num_batches_train):
    samples, x, y = d.GetBatch()
    eddl.train_batch(net, [x], [y])
    losses = eddl.get_losses(net)
    metrics = eddl.get_metrics(net)

    print(f'Train - epoch [{e + 1}/{args.epochs}] - batch [{b + 1}/{num_batches_train}]'
          f' - {loss_name}={losses[0]:.3f} - {metric_name}={metrics[0]:.3f}', flush=True)
d.Stop()
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

The DeepHealth Toolkit

- The DHt is composed by three main components:
  - The two libraries ECVL and EDDLL
  - The so called front-end.
- The front-end of the DHt is a software module that is divided into two parts, one visible to the user (a.k.a. graphical user interface–GUI) through the navigator (Firefox, Chrome, Edge, …) and one invisible part that performs all the actions indicated by the user using the functionalities provided by both libraries



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

The DeepHealth Toolkit

- The design of DeepHealth toolkit follows several core principles to support a set of key requirements:
  - Support a wide variety of application types in medical image analysis and computer-assisted diagnosis;
  - Be simple to be used in common use cases, but flexible enough for complex use cases;
  - Support parallel processing, model distribution and adaptation;
  - Support best practices as data augmentation and correct data set partitioning;
  - User-friendly visualization.

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

The DeepHealth Toolkit

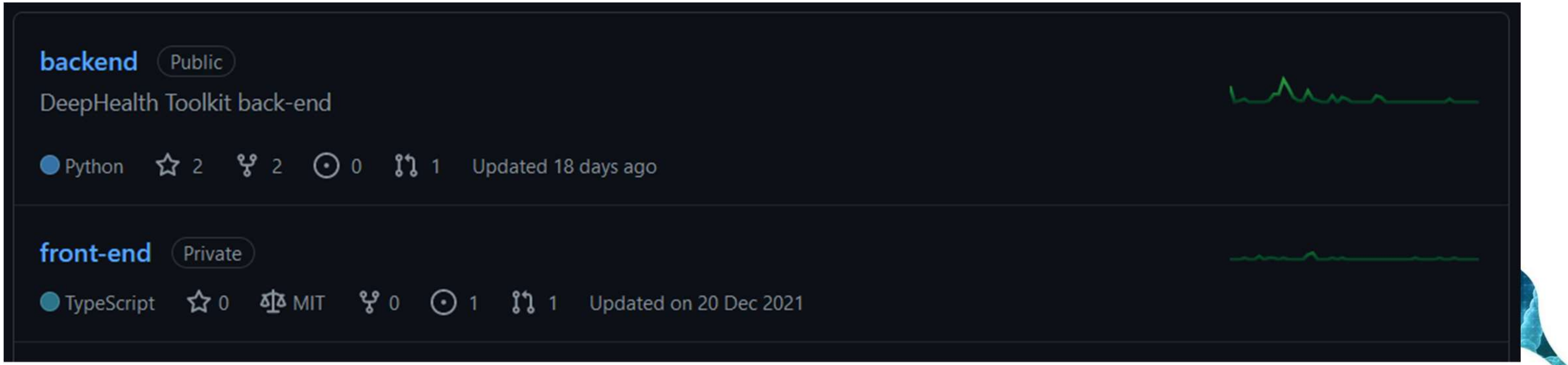
- The front-end allows user to access, in a graphic way, to all functionalities provided by the libraries (ECVL and EDDLL).
- It provides the graphical tools to exploit these functionalities in a user-friendly way. To achieve this goal, the GUI needs to be coupled with a back-end that will manage low level libraries interaction, driving the dataset loading, image manipulation and the training/inference processes.
- This architecture provides several benefits:

1. expert/end users do not have to manage individual software, installing and updating them at every library update;
2. similarly, the web browser interface will delegate the pairing with the hardware infrastructure to the software maintainer;
3. expert users will no longer have to worry about compatibility between the graphical user interface (GUI) and the server
4. The back-end includes the datasets management tool, and allows handling data/image manipulation pipelines.

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

The DeepHealth Toolkit

- The source code of both frontend and backend is available on the GitHub of the DeepHealth organization:
  - https://github.com/deephealthproject/backend
  - https://github.com/deephealthproject/front-end
- The corresponding dockerized versions are also available



backend Public

DeepHealth Toolkit back-end

● Python ☆ 2 ⑂ 2 ⊙ 0 ⇅ 1 Updated 18 days ago

front-end Private

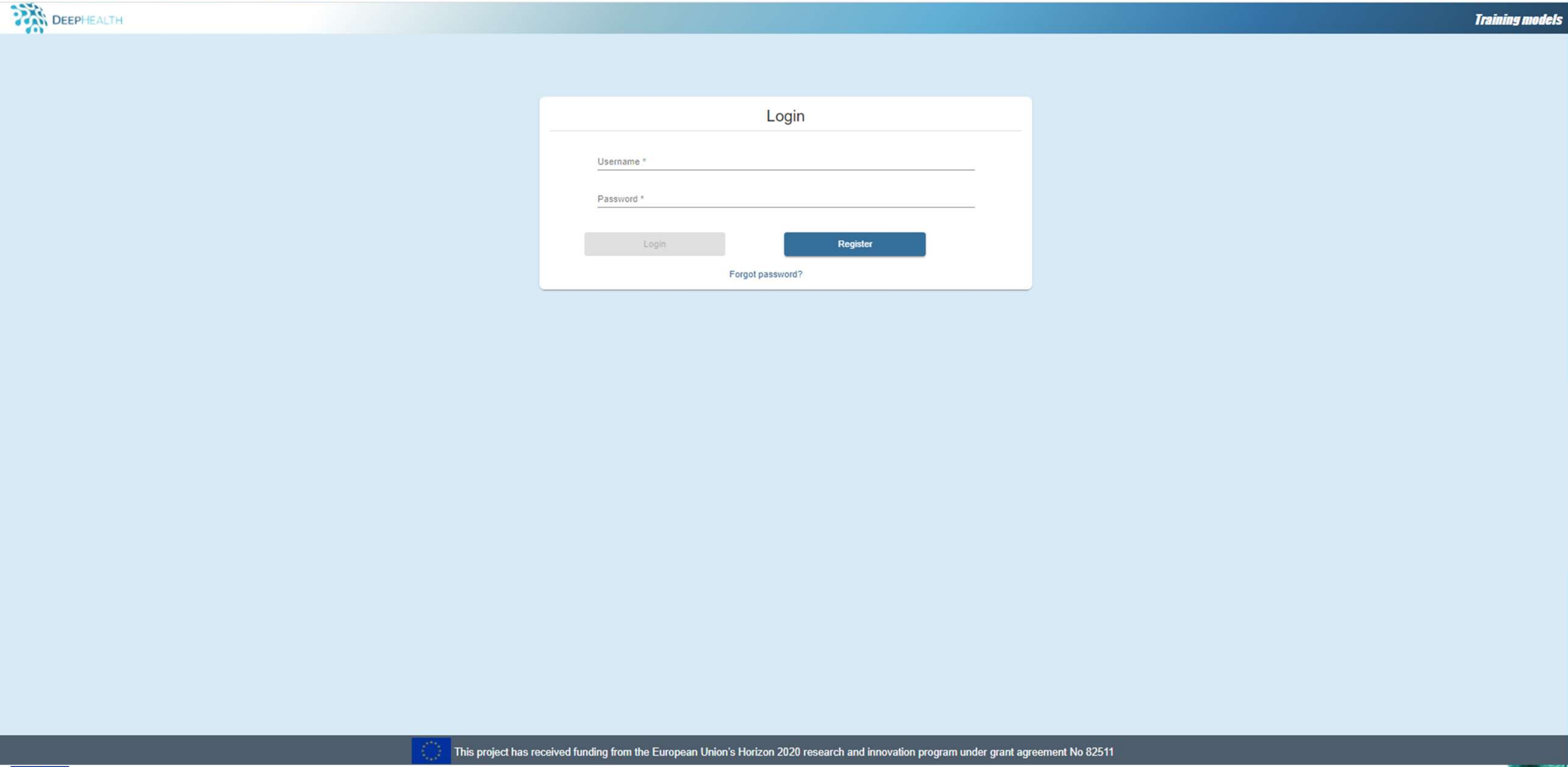
● TypeScript ☆ 0 ⚖ MIT ⑂ 0 ⊙ 1 ⇅ 1 Updated on 20 Dec 2021

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

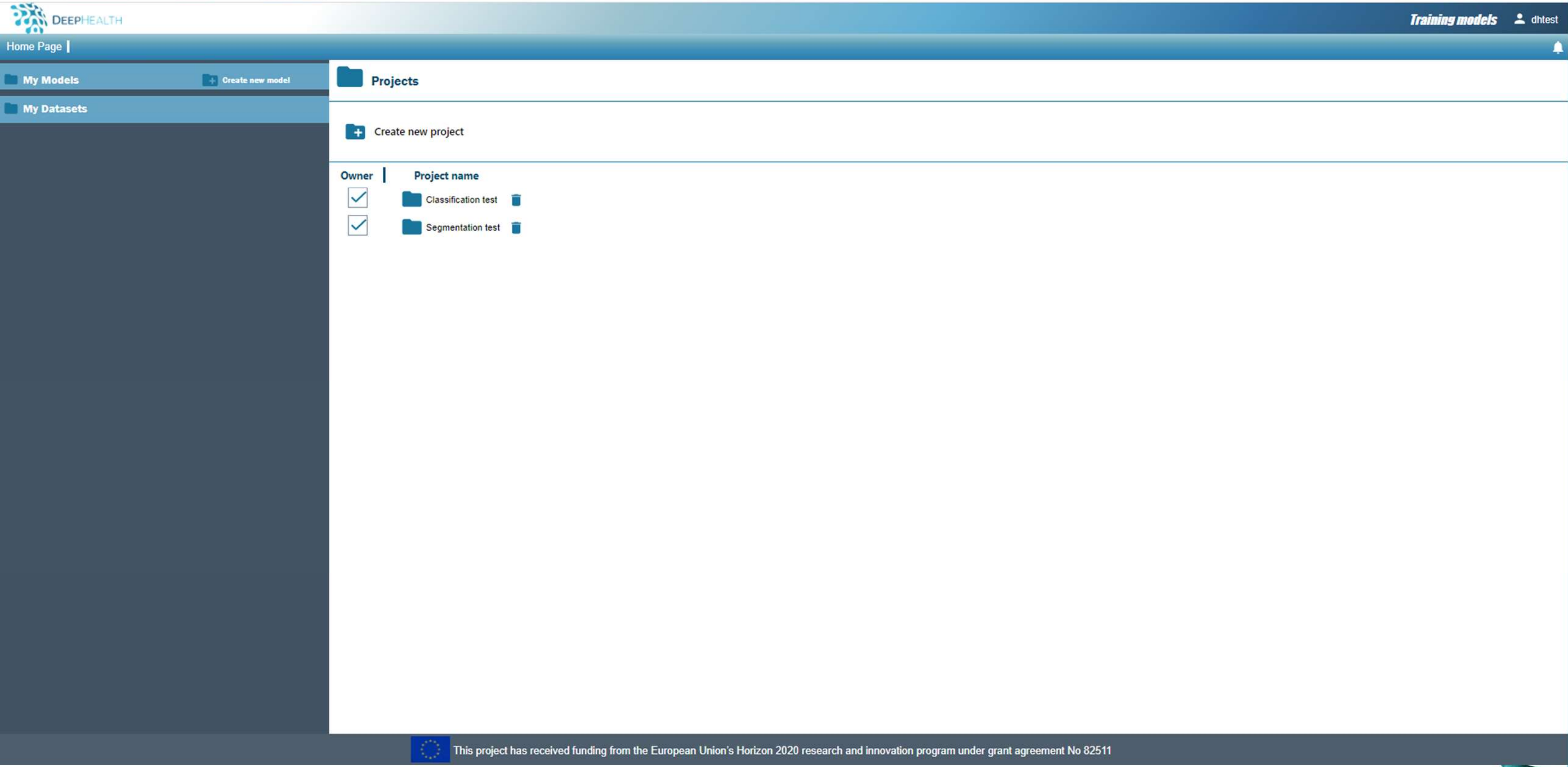
System Interaction and Use Case Scenarios

- The usage scenario is twofold:
  - Train: the user wants to see how well a model can perform on his data, by training it from scratch, or from an existing set of weights –a pretrained model– (e.g. learn to classify melanomas with resnet50 pre trained on ImageNet). This will create a new set of weights.
  - Inference: the user wants to see how well a model can perform on his data, by only running the data through a model with an existing set of weights. This does not create new weights for the selected model.

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

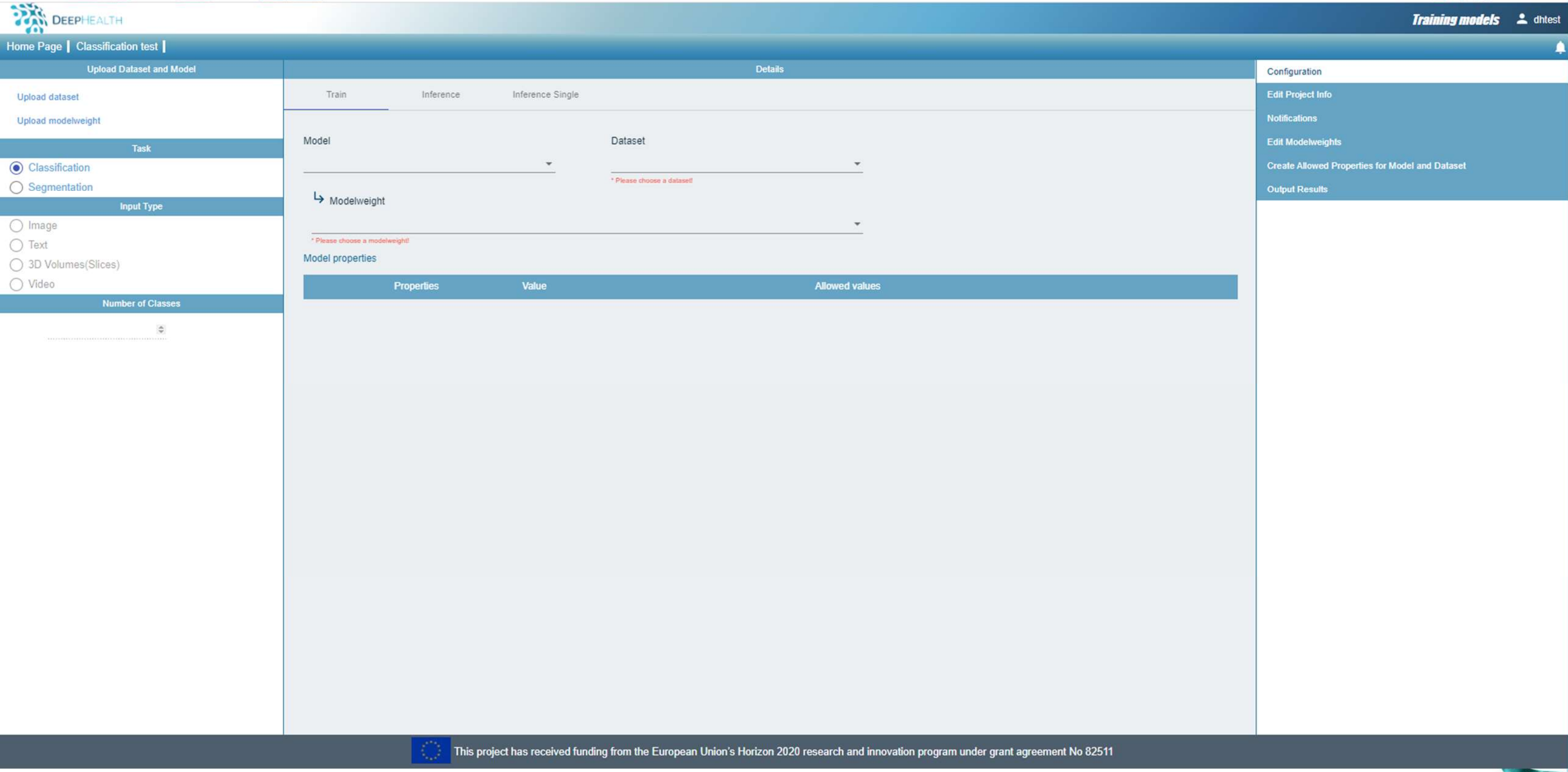


This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511



Owner	Project name
☑	Classification test
☑	Segmentation test

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511



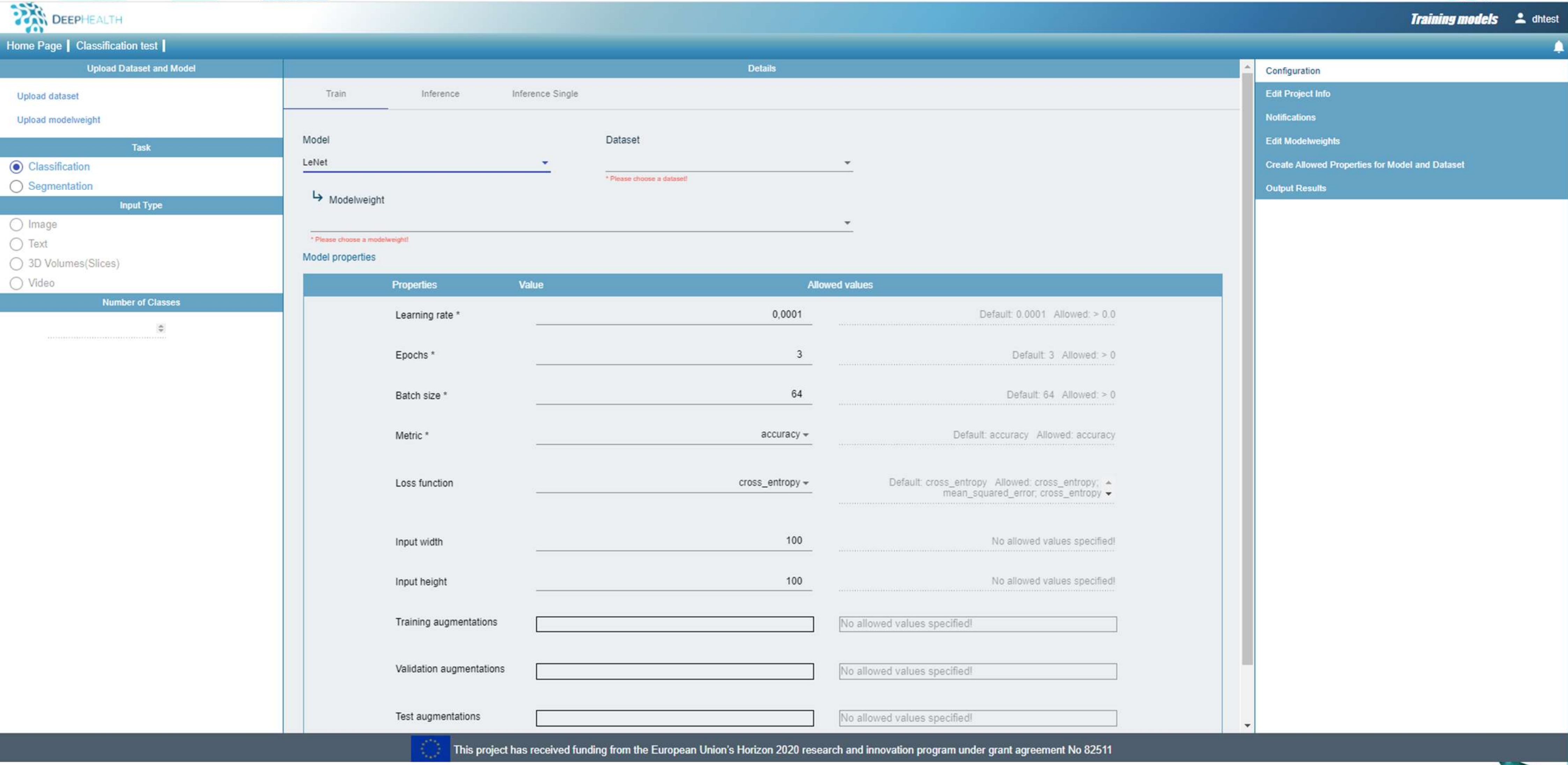
DEEPHEALTH

Training models dhtest

Home Page | Classification test |

Properties	Value	Allowed values

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511



Task: ● Classification ○ Segmentation

Input Type: ○ Image ○ Text ○ 3D Volumes(Slices) ○ Video

Model: LeNet

Dataset:

* Please choose a dataset!

↳ Modelweight

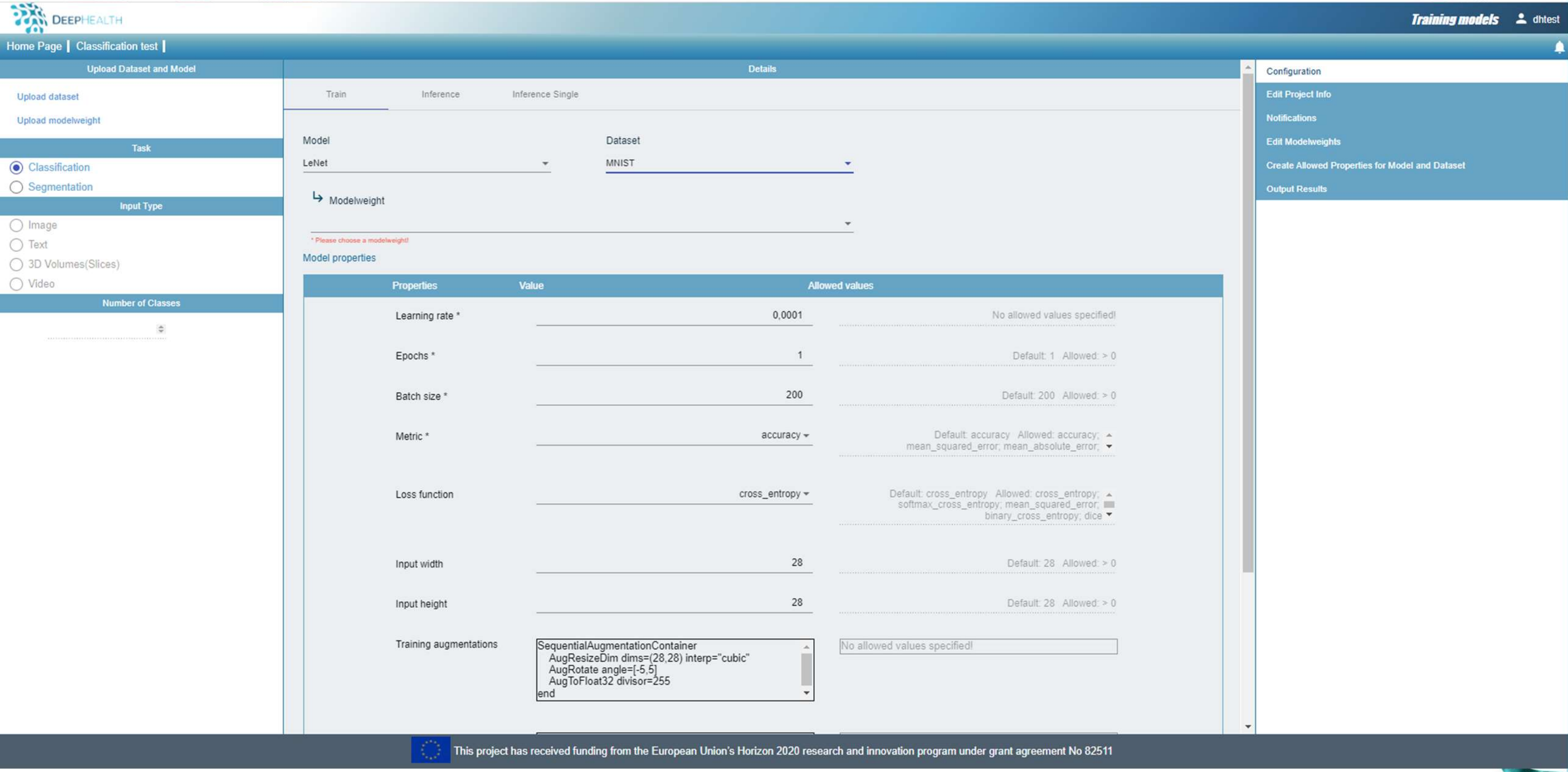
* Please choose a modelweight!

Model properties

Properties	Value	Allowed values
Learning rate *	0,0001	Default: 0.0001 Allowed: > 0.0
Epochs *	3	Default: 3 Allowed: > 0
Batch size *	64	Default: 64 Allowed: > 0
Metric *	accuracy	Default: accuracy Allowed: accuracy
Loss function	cross_entropy	Default: cross_entropy Allowed: cross_entropy; mean_squared_error; cross_entropy
Input width	100	No allowed values specified!
Input height	100	No allowed values specified!
Training augmentations		No allowed values specified!
Validation augmentations		No allowed values specified!
Test augmentations		No allowed values specified!

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.



Task

● Classification

○ Segmentation

Input Type

○ Image

○ Text

○ 3D Volumes(Slices)

○ Video

Number of Classes

Model: LeNet

Dataset: MNIST

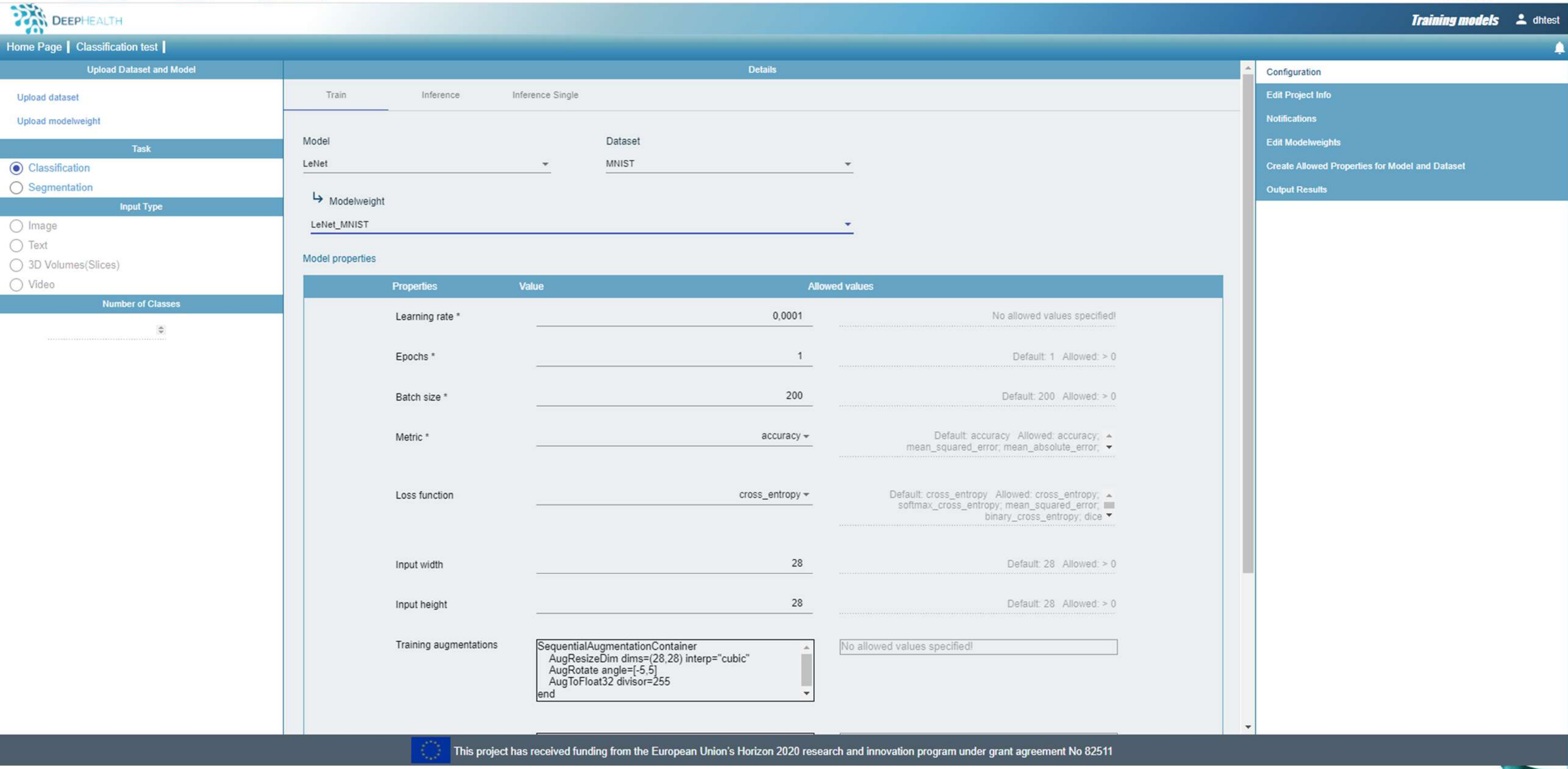
↳ Modelweight

* Please choose a modelweight!

Model properties

Properties	Value	Allowed values
Learning rate *	0,0001	No allowed values specified!
Epochs *	1	Default: 1 Allowed: > 0
Batch size *	200	Default: 200 Allowed: > 0
Metric *	accuracy	Default: accuracy Allowed: accuracy; mean_squared_error; mean_absolute_error;
Loss function	cross_entropy	Default: cross_entropy Allowed: cross_entropy; softmax_cross_entropy; mean_squared_error; binary_cross_entropy; dice
Input width	28	Default: 28 Allowed: > 0
Input height	28	Default: 28 Allowed: > 0
Training augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugRotate angle=[-5,5] AugToFloat32 divisor=255 end	No allowed values specified!

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511


DEEPHEALTH

Training models

dhtest

Home Page | Classification test |

Upload Dataset and Model

Upload dataset

Upload modelweight

Task

● Classification

○ Segmentation

Input Type

○ Image

○ Text

○ 3D Volumes(Slices)

○ Video

Number of Classes

Details

Train | Inference | Inference Single

Model: LeNet

Dataset: MNIST

↳ Modelweight: LeNet_MNIST

Model properties

Properties	Value	Allowed values
Learning rate *	0,0001	No allowed values specified!
Epochs *	1	Default: 1 Allowed: > 0
Batch size *	200	Default: 200 Allowed: > 0
Metric *	accuracy	Default: accuracy Allowed: accuracy; mean_squared_error; mean_absolute_error;
Loss function	cross_entropy	Default: cross_entropy Allowed: cross_entropy; softmax_cross_entropy; mean_squared_error; binary_cross_entropy; dice
Input width	28	Default: 28 Allowed: > 0
Input height	28	Default: 28 Allowed: > 0
Training augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugRotate angle=[-5,5] AugToFloat32 divisor=255 end	No allowed values specified!

Configuration

Edit Project Info

Notifications

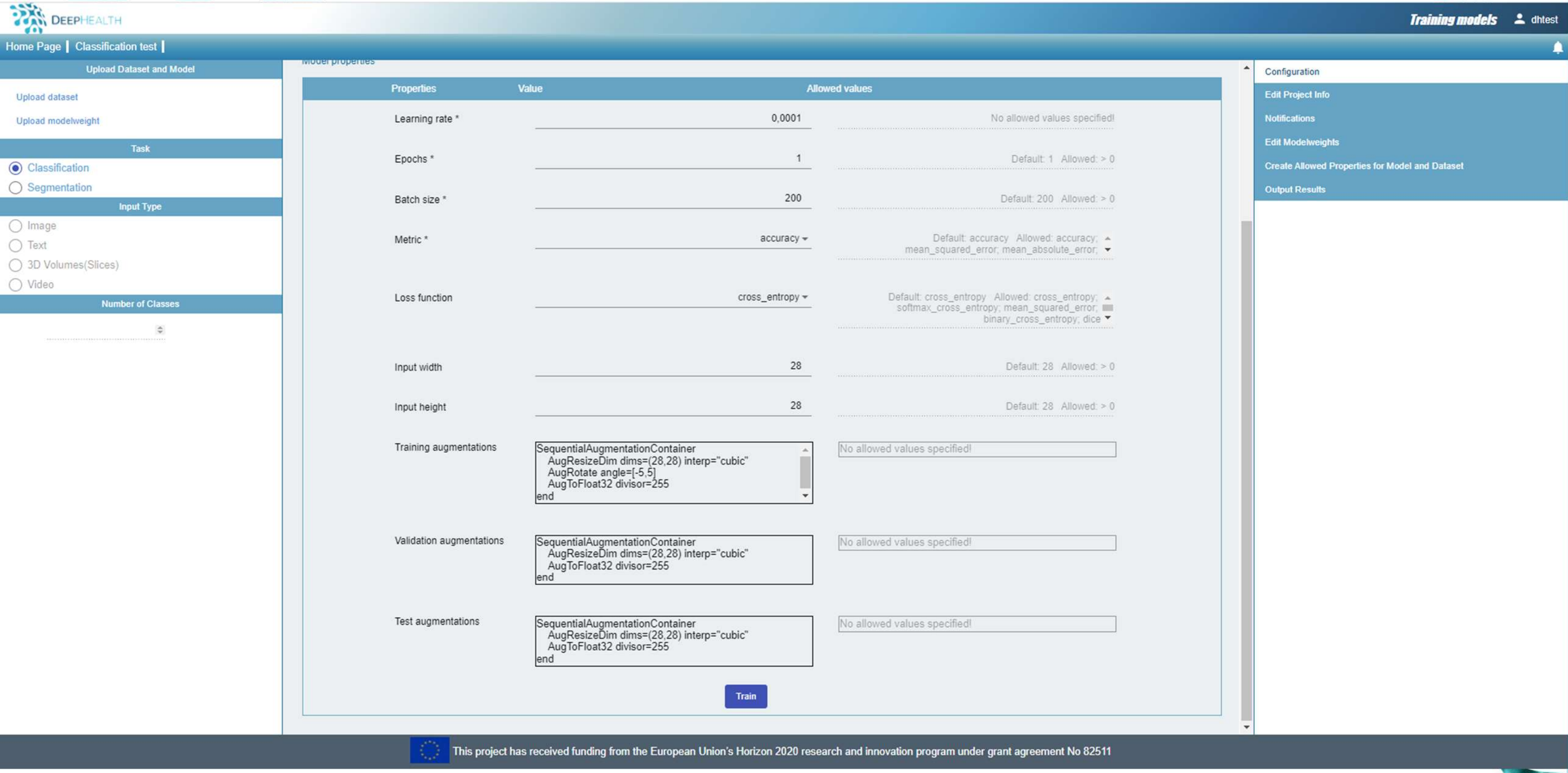
Edit Modelweights

Create Allowed Properties for Model and Dataset

Output Results

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

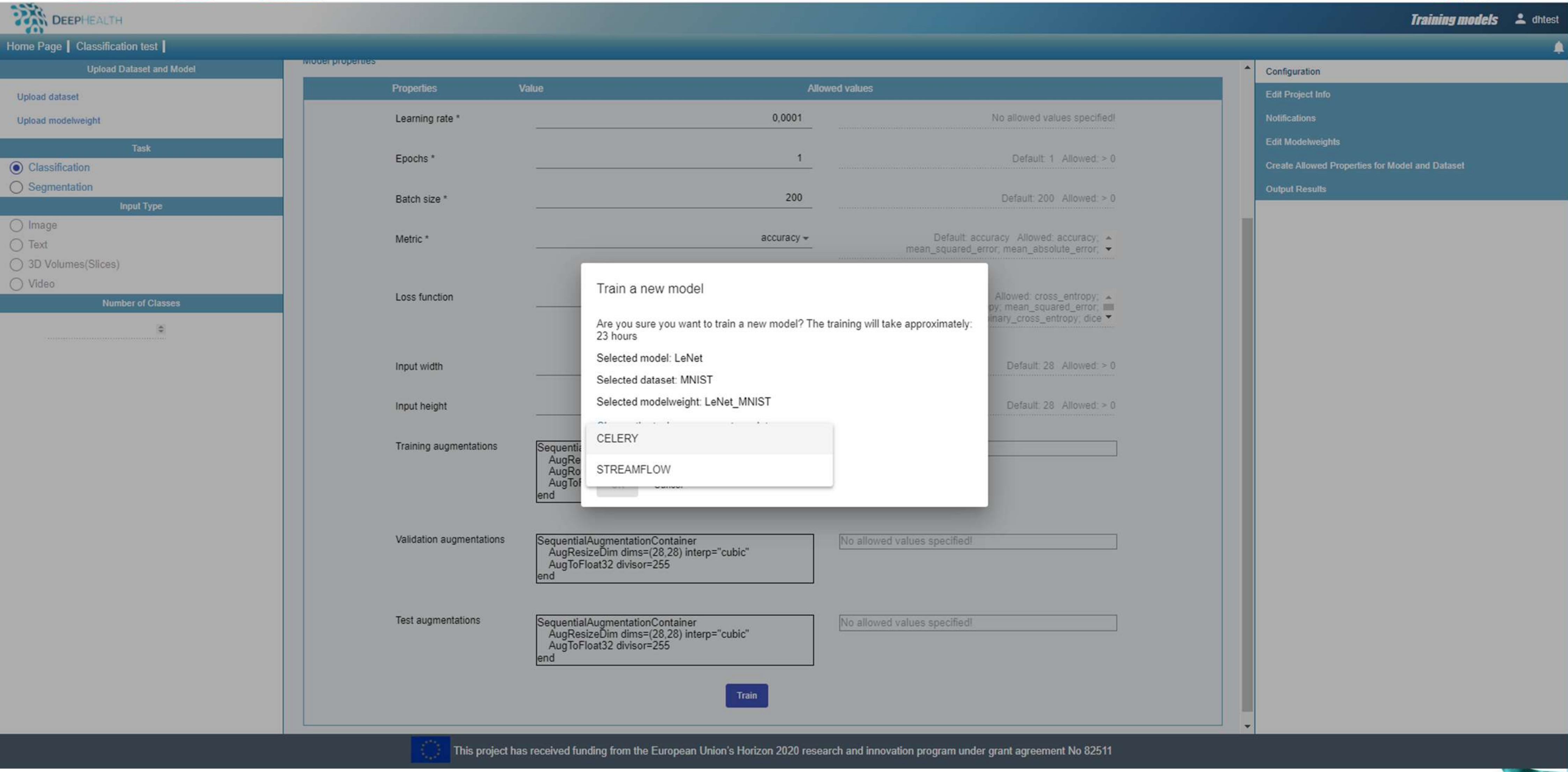


Task: ● Classification ○ Segmentation

Input Type: ○ Image ○ Text ○ 3D Volumes(Slices) ○ Video

Properties	Value	Allowed values
Learning rate *	0,0001	No allowed values specified!
Epochs *	1	Default: 1 Allowed: > 0
Batch size *	200	Default: 200 Allowed: > 0
Metric *	accuracy	Default: accuracy Allowed: accuracy; mean_squared_error; mean_absolute_error;
Loss function	cross_entropy	Default: cross_entropy Allowed: cross_entropy; softmax_cross_entropy; mean_squared_error; binary_cross_entropy; dice
Input width	28	Default: 28 Allowed: > 0
Input height	28	Default: 28 Allowed: > 0
Training augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugRotate angle=[-5,5] AugToFloat32 divisor=255 end	No allowed values specified!
Validation augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugToFloat32 divisor=255 end	No allowed values specified!
Test augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugToFloat32 divisor=255 end	No allowed values specified!

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511



Task

● Classification
○ Segmentation

Input Type

○ Image
○ Text
○ 3D Volumes(Slices)
○ Video

Number of Classes

Properties	Value	Allowed values
Learning rate *	0,0001	No allowed values specified!
Epochs *	1	Default: 1 Allowed: > 0
Batch size *	200	Default: 200 Allowed: > 0
Metric *	accuracy	Default: accuracy Allowed: accuracy; mean_squared_error; mean_absolute_error;
Loss function		Allowed: cross_entropy; py; mean_squared_error; inary_cross_entropy; dice
Input width		Default: 28 Allowed: > 0
Input height		Default: 28 Allowed: > 0
Training augmentations	Sequenti AugRe AugRo AugTo end	
Validation augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugToFloat32 divisor=255 end	No allowed values specified!
Test augmentations	SequentialAugmentationContainer AugResizeDim dims=(28,28) interp="cubic" AugToFloat32 divisor=255 end	No allowed values specified!

Train a new model

Are you sure you want to train a new model? The training will take approximately: 23 hours

Selected model: LeNet

Selected dataset: MNIST

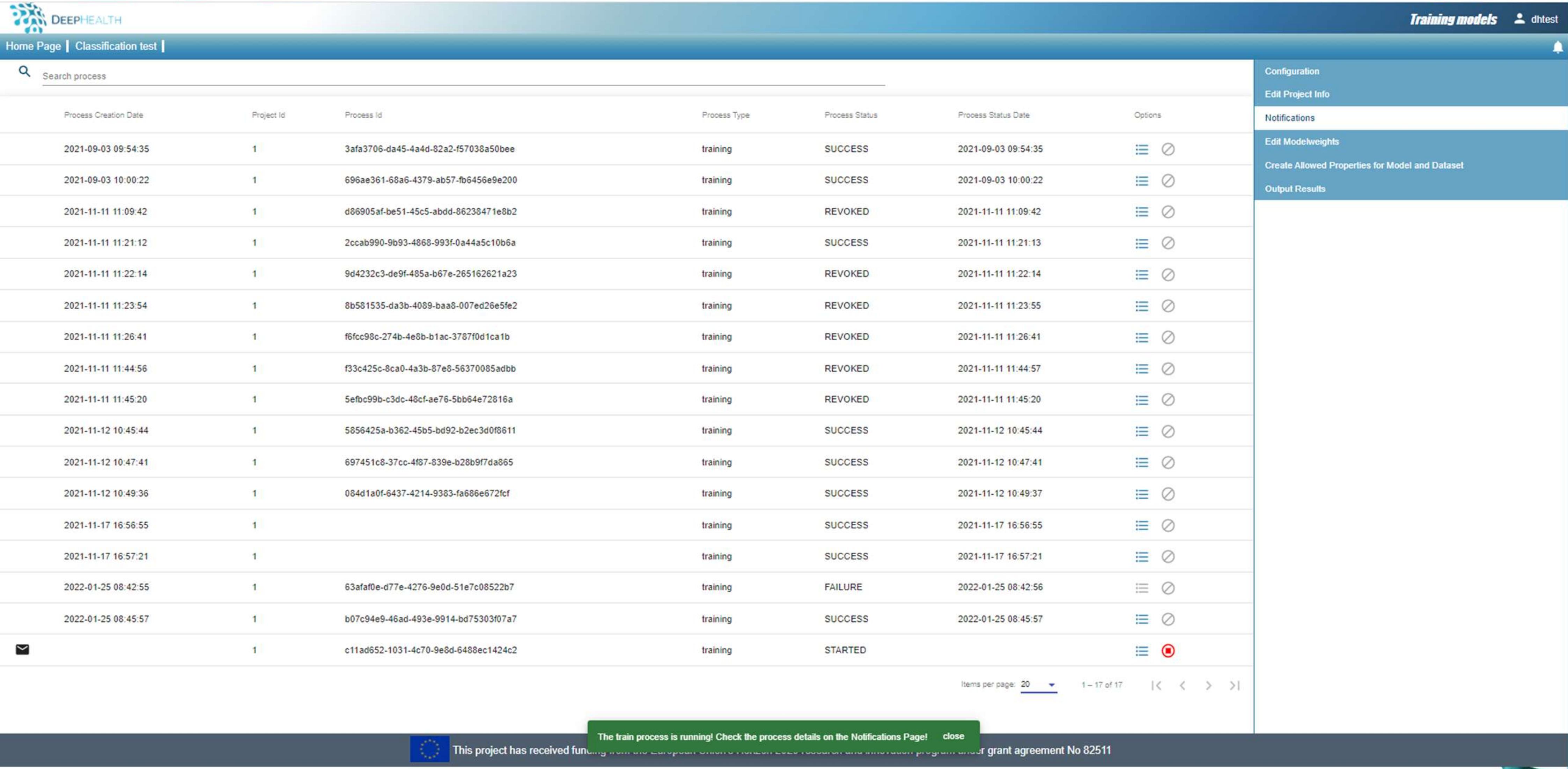
Selected modelweight: LeNet_MNIST

CELERY

STREAMFLOW

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

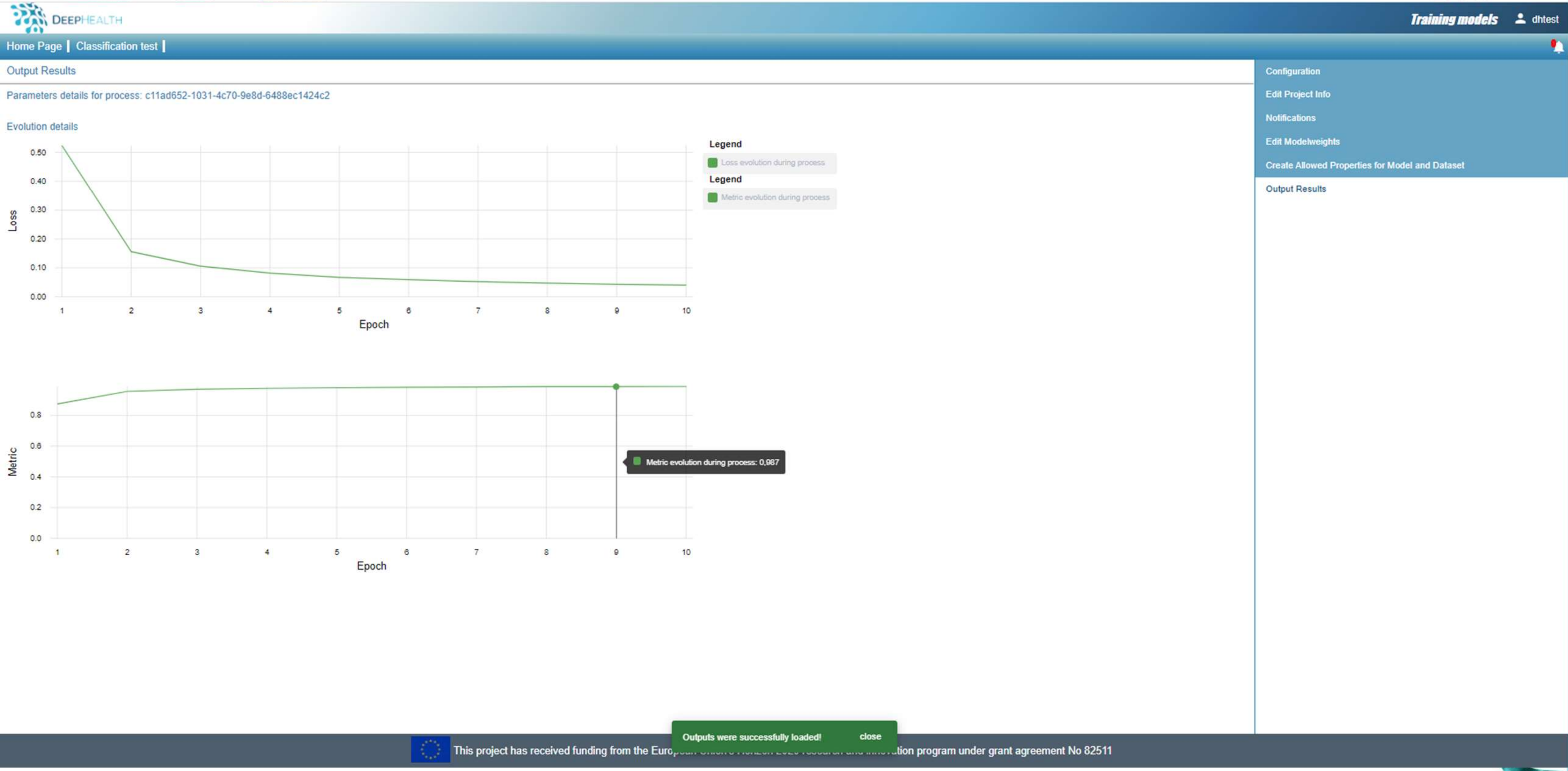


	Process Creation Date	Project Id	Process Id	Process Type	Process Status	Process Status Date	Options
	2021-09-03 09:54:35	1	3afa3706-da45-4a4d-82a2-f57038a50bee	training	SUCCESS	2021-09-03 09:54:35	☰ ⊘
	2021-09-03 10:00:22	1	696ae361-68a6-4379-ab57-fb6456e9e200	training	SUCCESS	2021-09-03 10:00:22	☰ ⊘
	2021-11-11 11:09:42	1	d86905af-be51-45c5-abdd-86238471e8b2	training	REVOKED	2021-11-11 11:09:42	☰ ⊘
	2021-11-11 11:21:12	1	2ccab990-9b93-4868-993f-0a44a5c10b6a	training	SUCCESS	2021-11-11 11:21:13	☰ ⊘
	2021-11-11 11:22:14	1	9d4232c3-de9f-485a-b67e-265162621a23	training	REVOKED	2021-11-11 11:22:14	☰ ⊘
	2021-11-11 11:23:54	1	8b581535-da3b-4089-baa8-007ed26e5fe2	training	REVOKED	2021-11-11 11:23:55	☰ ⊘
	2021-11-11 11:26:41	1	f6fcc98c-274b-4e8b-b1ac-3787f0d1ca1b	training	REVOKED	2021-11-11 11:26:41	☰ ⊘
	2021-11-11 11:44:56	1	f33c425c-8ca0-4a3b-87e8-56370085adbb	training	REVOKED	2021-11-11 11:44:57	☰ ⊘
	2021-11-11 11:45:20	1	5efbc99b-c3dc-48cf-ae76-5bb64e72816a	training	REVOKED	2021-11-11 11:45:20	☰ ⊘
	2021-11-12 10:45:44	1	5856425a-b362-45b5-bd92-b2ec3d0f8611	training	SUCCESS	2021-11-12 10:45:44	☰ ⊘
	2021-11-12 10:47:41	1	697451c8-37cc-4f87-839e-b28b9f7da865	training	SUCCESS	2021-11-12 10:47:41	☰ ⊘
	2021-11-12 10:49:36	1	084d1a0f-6437-4214-9383-fa686e672fcf	training	SUCCESS	2021-11-12 10:49:37	☰ ⊘
	2021-11-17 16:56:55	1		training	SUCCESS	2021-11-17 16:56:55	☰ ⊘
	2021-11-17 16:57:21	1		training	SUCCESS	2021-11-17 16:57:21	☰ ⊘
	2022-01-25 08:42:55	1	63afaf0e-d77e-4276-9e0d-51e7c08522b7	training	FAILURE	2022-01-25 08:42:56	☰ ⊘
	2022-01-25 08:45:57	1	b07c94e9-46ad-493e-9914-bd75303f07a7	training	SUCCESS	2022-01-25 08:45:57	☰ ⊘
✉		1	c11ad652-1031-4c70-9e8d-6488ec1424c2	training	STARTED		☰ ●

Items per page: 20 1 – 17 of 17

The train process is running! Check the process details on the Notifications Page! close

This project has received fun[illegible]r grant agreement No 82511



Output Results

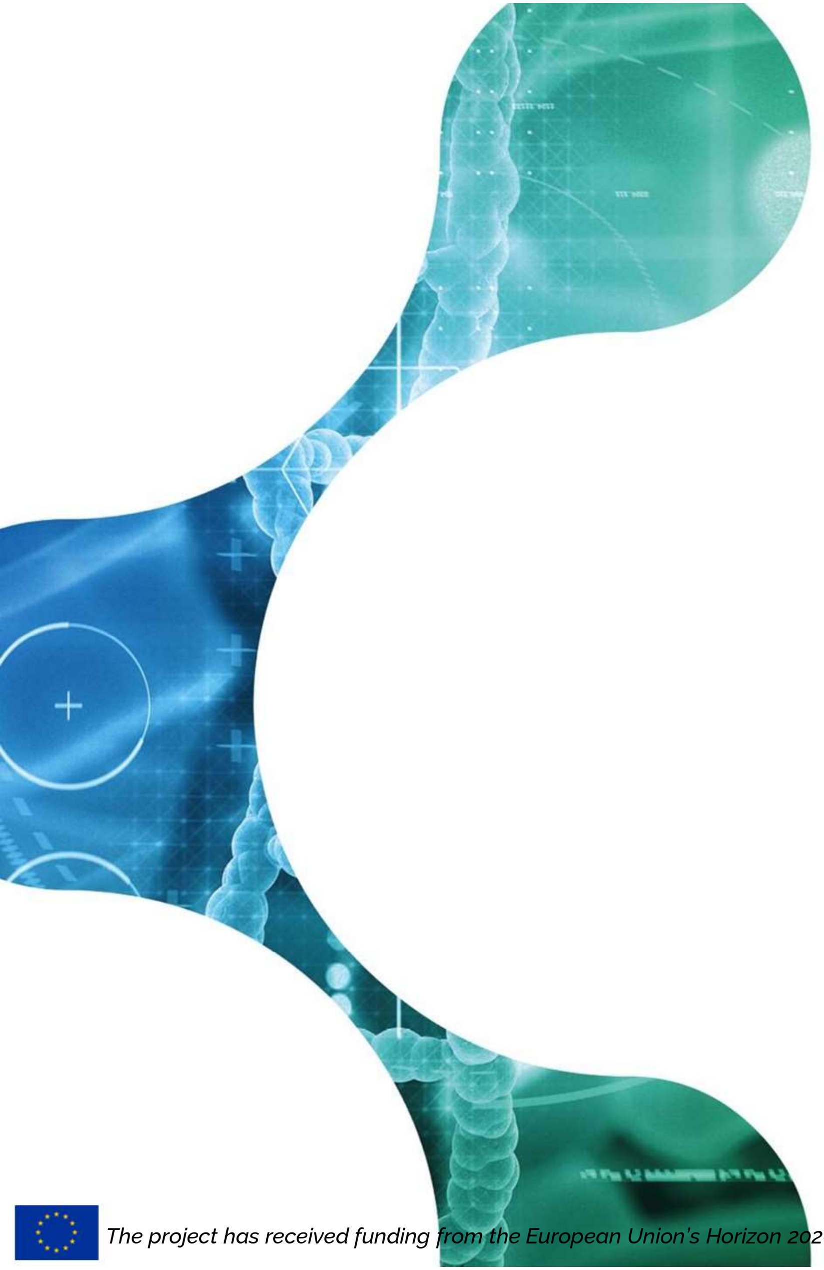
Parameters details for process: c11ad652-1031-4c70-9e8d-6488ec1424c2

Evolution details

🟩 Metric evolution during process: 0,987

Outputs were successfully loaded!

This project has received funding from the Euro ... tion program under grant agreement No 82511



DEEPHEALTH

Thank you!

Costantino Grana
costantino.grana@unimore.it

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.