MINERVA

Sep 09, 2020

User Guides

1 Installation Guide **1**
- 1.1 Recommended Platforms . . . 1
- 1.2 Install MINERVA . . . 1
- 1.3 Install MINERVA via Docker . . . 2

2 Getting Started **3**
- 2.1 Prepare Datasets . . . 3
- 2.2 Start Server . . . 4
- 2.3 Upload Dataset . . . 5
- 2.4 Create Project . . . 8
- 2.5 Start Training . . . 8
- 2.6 Export Policy Function . . . 11

3 Tutorials **13**
- 3.1 CartPole . . . 13

4 MINERVA CLI **15**
- 4.1 run . . . 15
- 4.2 clean . . . 15
- 4.3 upgrade-db . . . 15
- 4.4 downgrade-db . . . 15

5 License **17**

6 Indices and tables **19**

CHAPTER 1

Installation Guide

1.1 Recommended Platforms

1.1.1 Operating System

MINERVA is only tested on Linux and macOS. However, you can possibly run MINERVA on Windows as long as PyTorch runs since it's the core dependency.

1.1.2 Browser

For now, MINERVA is only tested with Chrome. There would be incompatibilities with other browsers (I've confirmed some glitches on Safari).

1.2 Install MINERVA

1.2.1 Install MINERVA via PyPI

pip is a recommended way to install minerva:

```
$ pip install minerva-ui
```

1.2.2 Install MINERVA from source

You can also install via GitHub repository:

```
$ git clone https://github.com/takuseno/minerva
$ cd minerva
$ npm install
$ npm run build
$ pip install -e .
```

1.3 Install MINERVA via Docker

If you use GPU devices, you need to setup nvidia-docker properly:

```
$ docker run -d --gpus all -p 9000:9000 --name minerva takuseno/minerva:latest
$ # MINERVA server is running
```

CHAPTER 2

Getting Started

2.1 Prepare Datasets

2.1.1 Dataset with vector observation

The dataset must be a CSV file containing the following columns. The data should be chronologically ordered.

Table 1: columns

column name	description
episode	an episode ID
observation:X	a real value for the Xth dimension in an observation
action:X	a real value for the Xth dimension in an action (continuous control) or an action ID (discrete control)
reward	a real value for reward

This is an example CartPole data:

```
episode,observation:0,observation:1,observation:2,observation:3,action:0,reward
0,0.03197332076282214,0.023978136772313002,-0.01460231690901137,0.01428123941035453,1,
↪0.0
0,0.0324528834982684,0.21930642661209865,-0.0143166921208042 8,-0.28297288746447075,0,
↪1.0
.
.
.
```

2.1.2 Dataset with image observation

The dataset must contain a CSV file and image files. The data must contain the following columns.

Table 2: columns

column name	description
episode	an episode ID
observation:0	a image file name (e.g. observation_0.png)
action:X	a real value for the Xth dimension in an action (continuous control) or an action ID (discrete control)
reward	a real value for reward

This is an example:

```
episode,observation:0,action:0,reward
0,observation_0.png,1,0.0
0,observation_1.png,0,0.0
.
.
.
```

Note: The image files must be located in the directory that contains only image files to upload.

2.2 Start Server

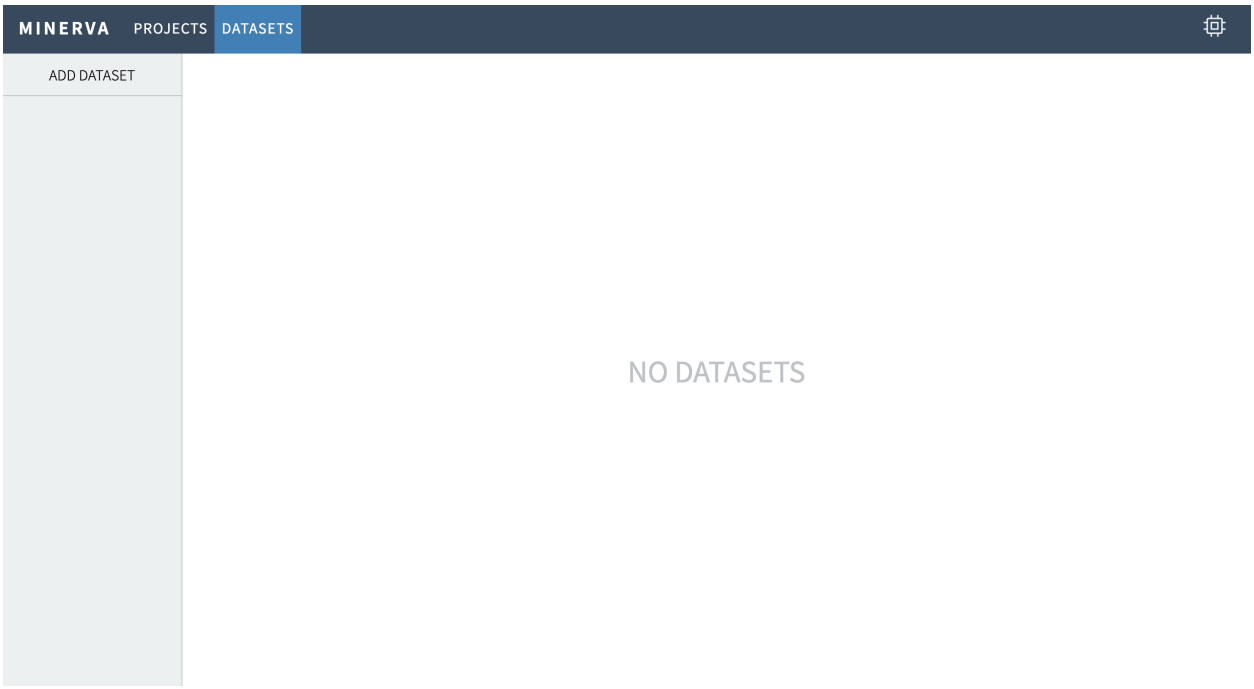
At the first launch, `$HOME/.minerva` will be created to store datasets, databases and training metrics. You can configure this by setting `$MINERVA_DIR`. For example:

```
$ export MINERVA_DIR=$HOME/.custom_dir
```

Now you can start MINERVA as follows:

```
$ minerva run [--host HOST_NAME] [--port PORT]
```

Then, open http://localhost:9000 and you'll see the MINERVA UI.

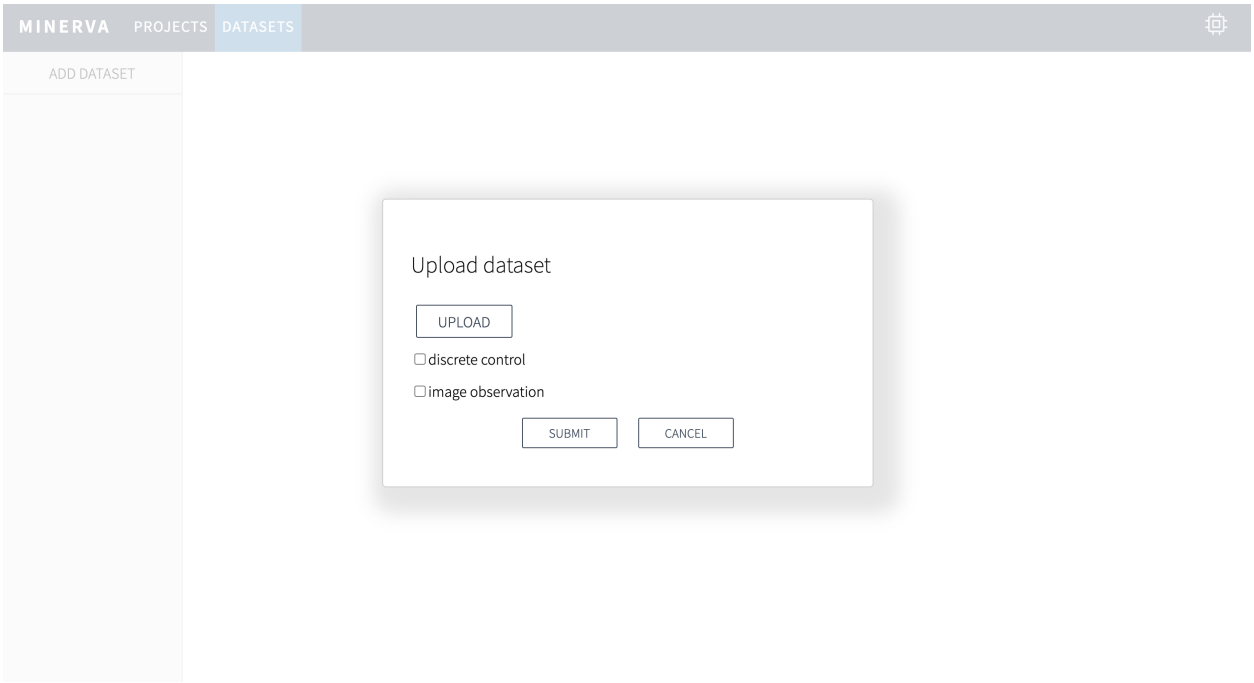


2.3 Upload Dataset

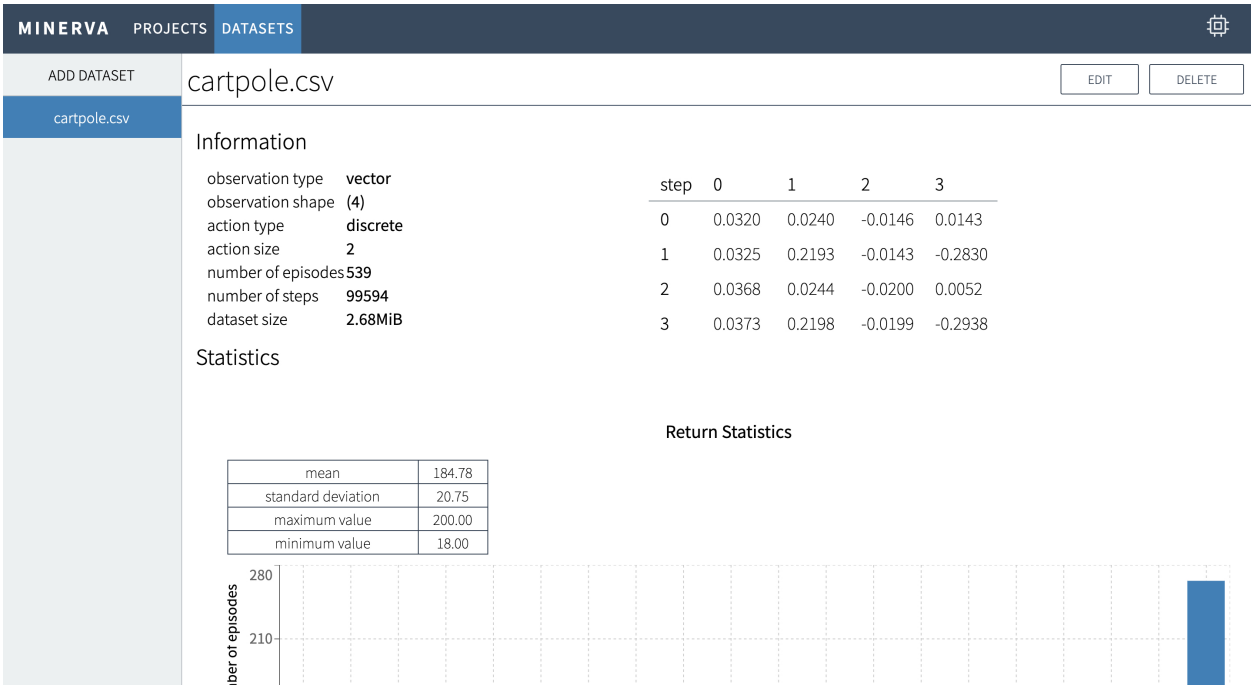
To upload a new dataset, click `ADD DATASET` button.

2.3.1 Upload dataset with vector observation

1. Click `UPLOAD` button to select the dataset CSV file.
2. Check `discrete control` if the action-space is discrete.
3. Click `SUBMIT` to upload the dataset.



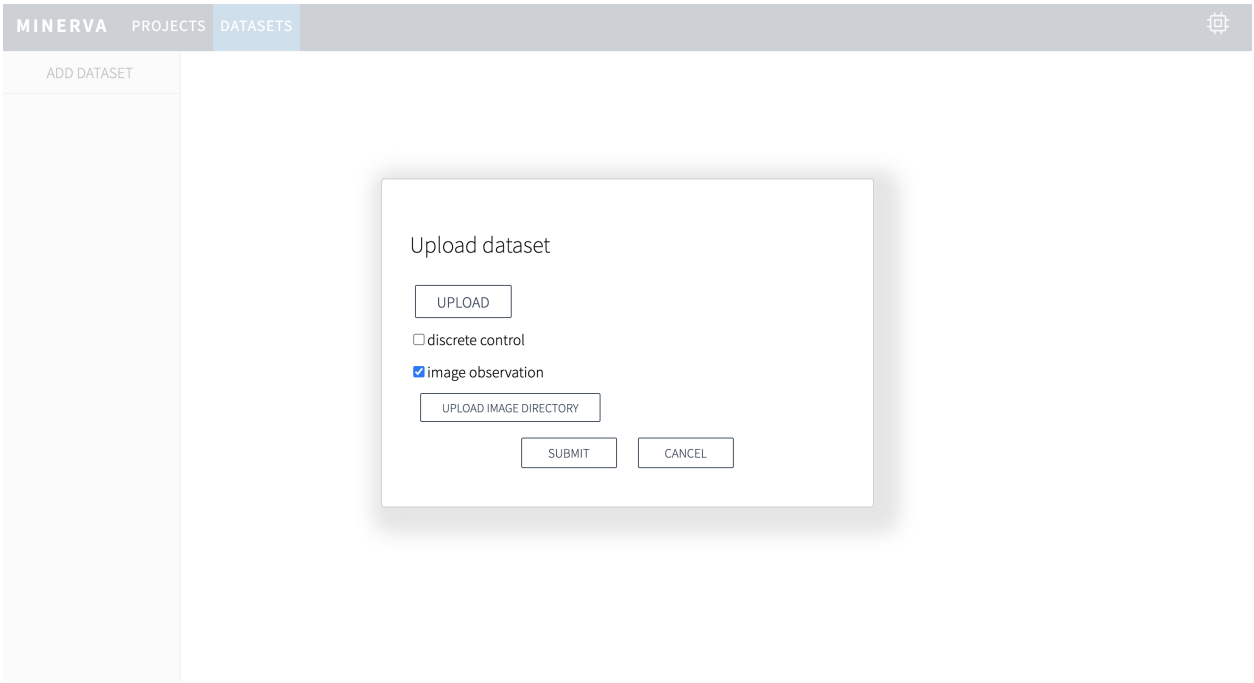
This is an example dashboard screen after uploading a vector dataset.



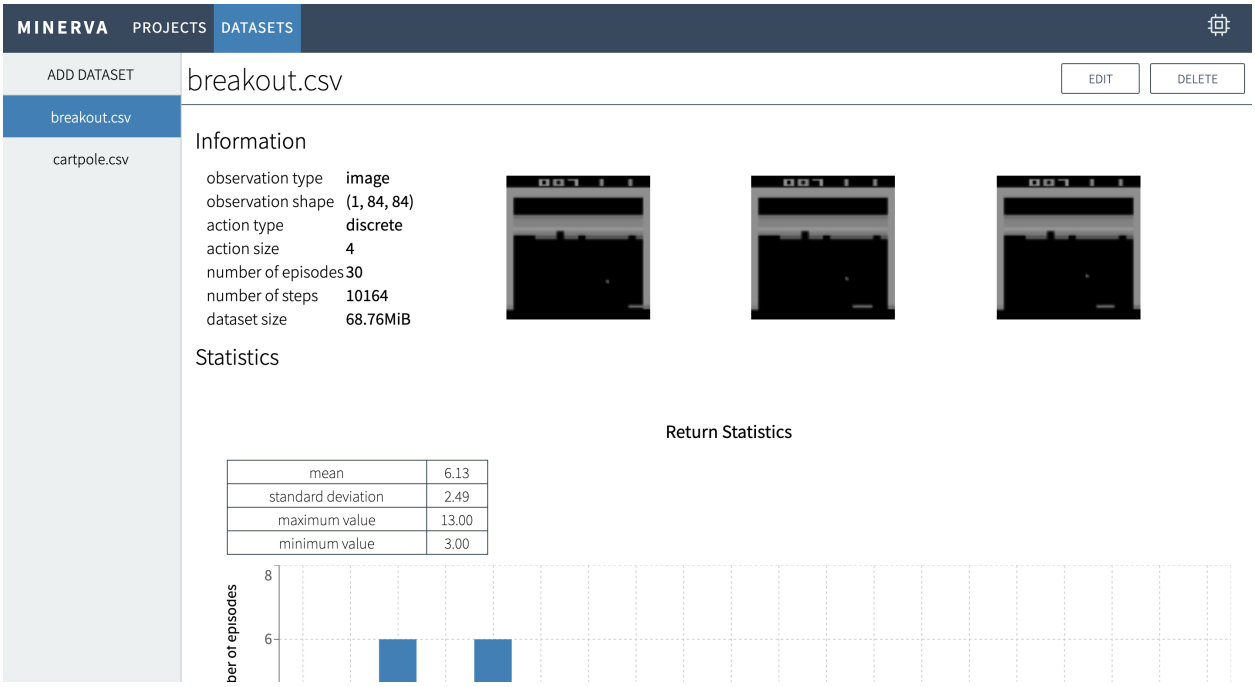
2.3.2 Upload dataset with image observation

1. Click `UPLOAD` button to select the dataset CSV file.
2. Check `discrete control` if the action-space is discrete.
3. Check `image observation`.
4. Click `UPLOAD IMAGE DIRECTORY` button to select the directory containing image files.

5. Click `SUBMIT` to upload the dataset.



This is an example dashboard screen after uploading an image dataset.



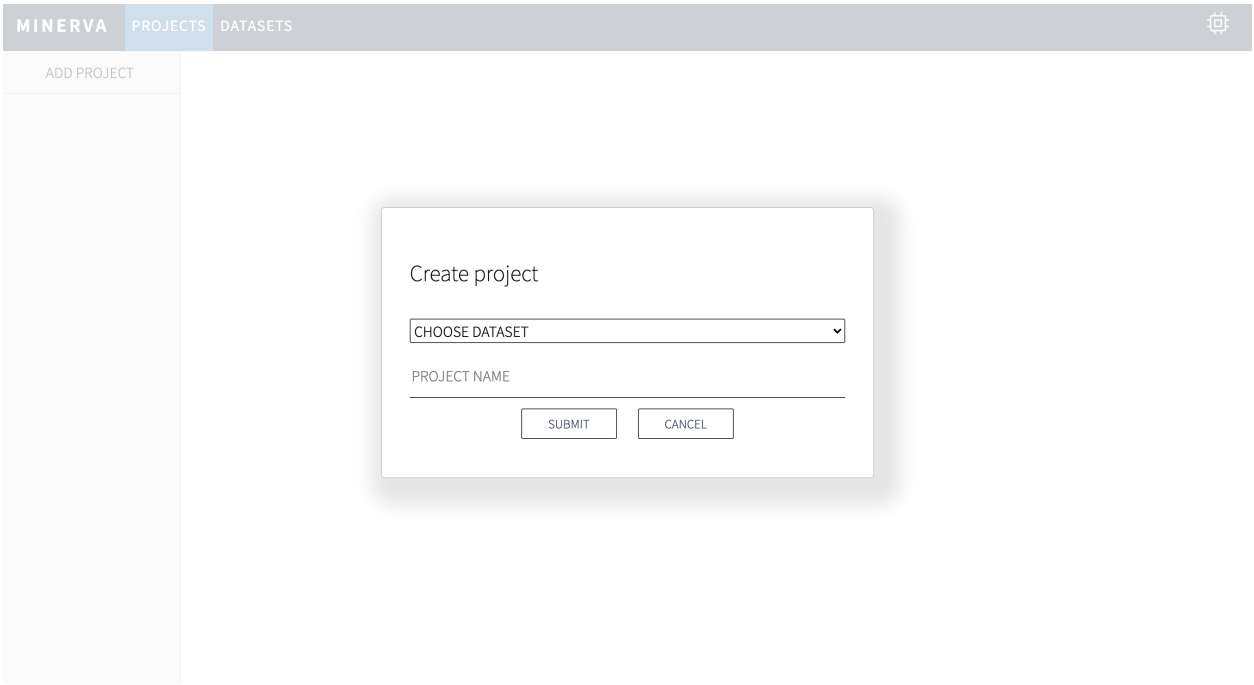
Note: The all files in the selected directory will be uploaded.

2.4 Create Project

To create a new project, click `ADD PROJECT` in the project page.

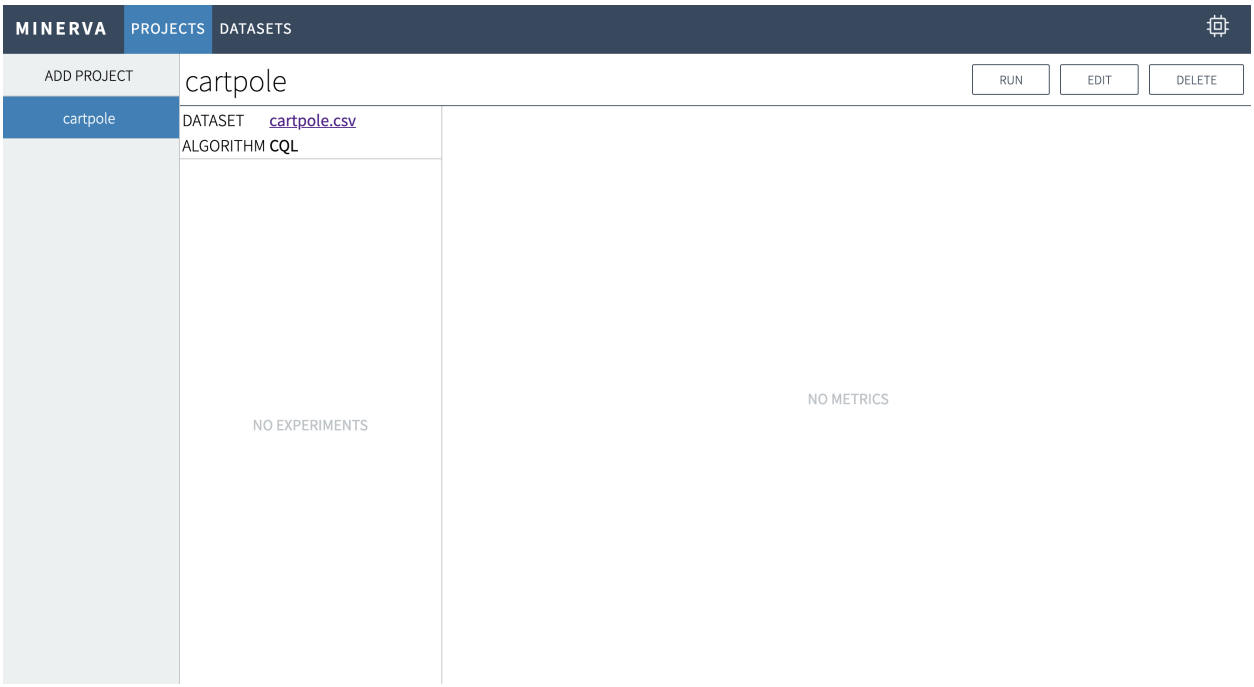
Then,

1. Choose a dataset from the uploaded ones.
2. Fill the project name.
3. Click `SUBMIT` button to create.

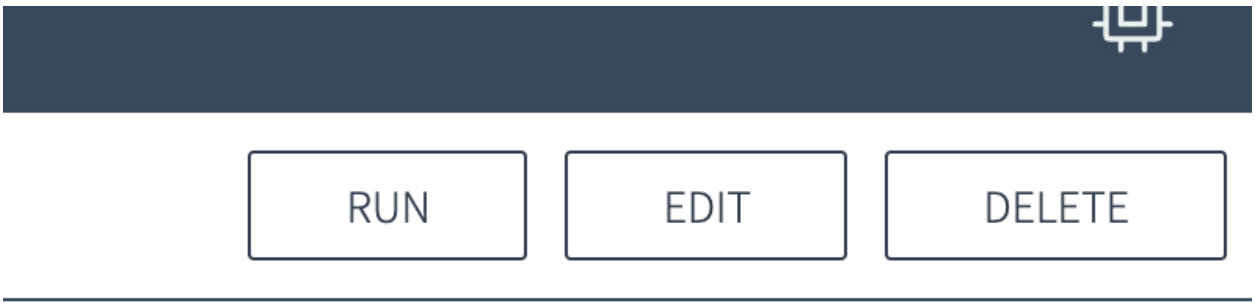


2.5 Start Training

Once you created a project, you will see an empty project like below.

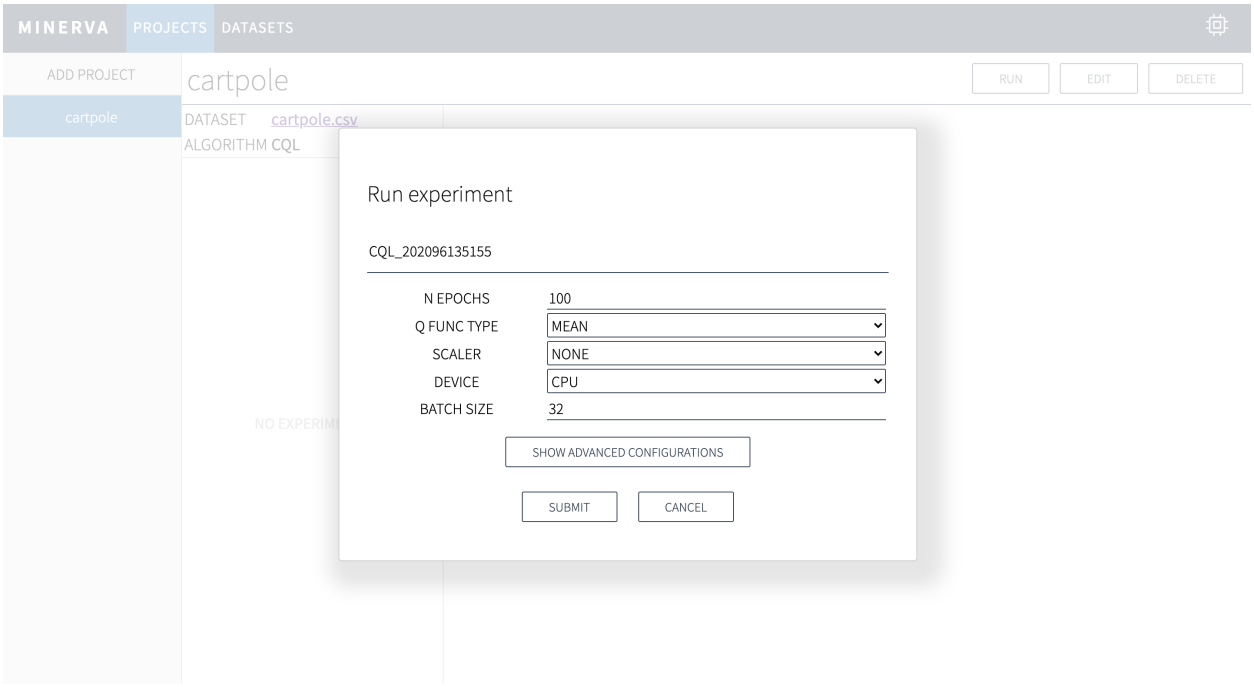


Click `RUN` button to start training.



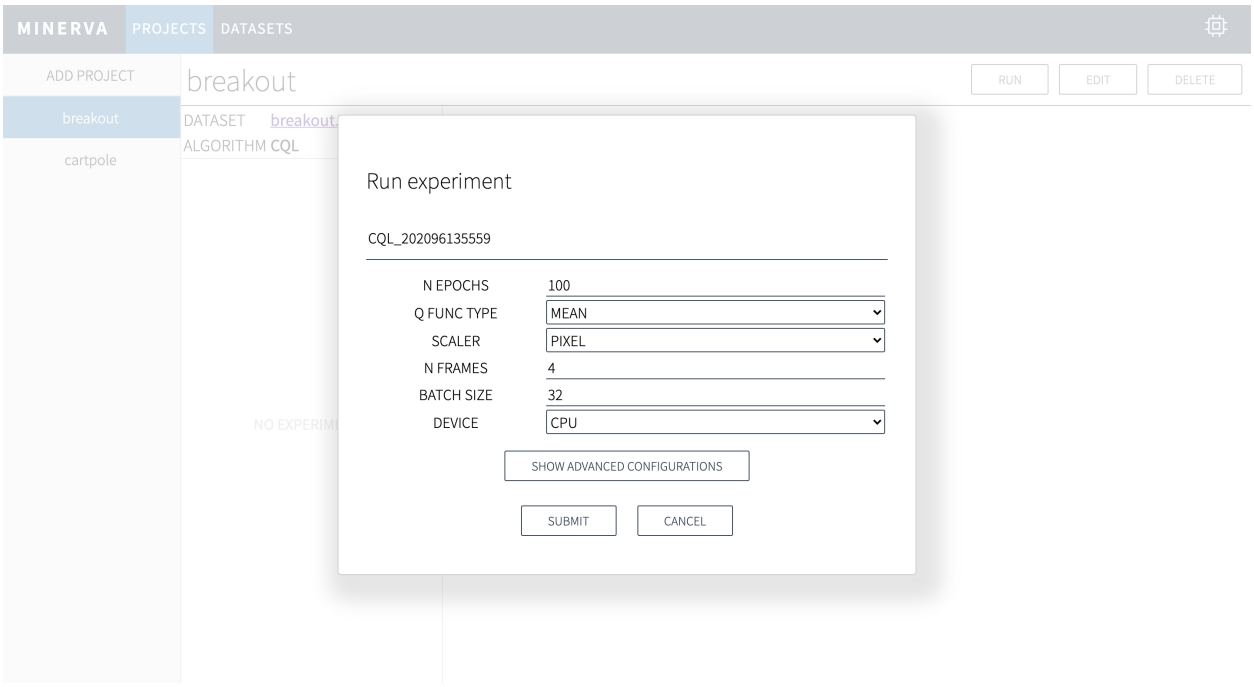
2.5.1 Train with vector observation

1. Configure training settings.
2. Choose device to use CPU or GPU.
3. (optional) Configure advanced settings to click `SHOW ADVANCED CONFIGURATIONS`.
4. Click `SUBMIT` to start training.



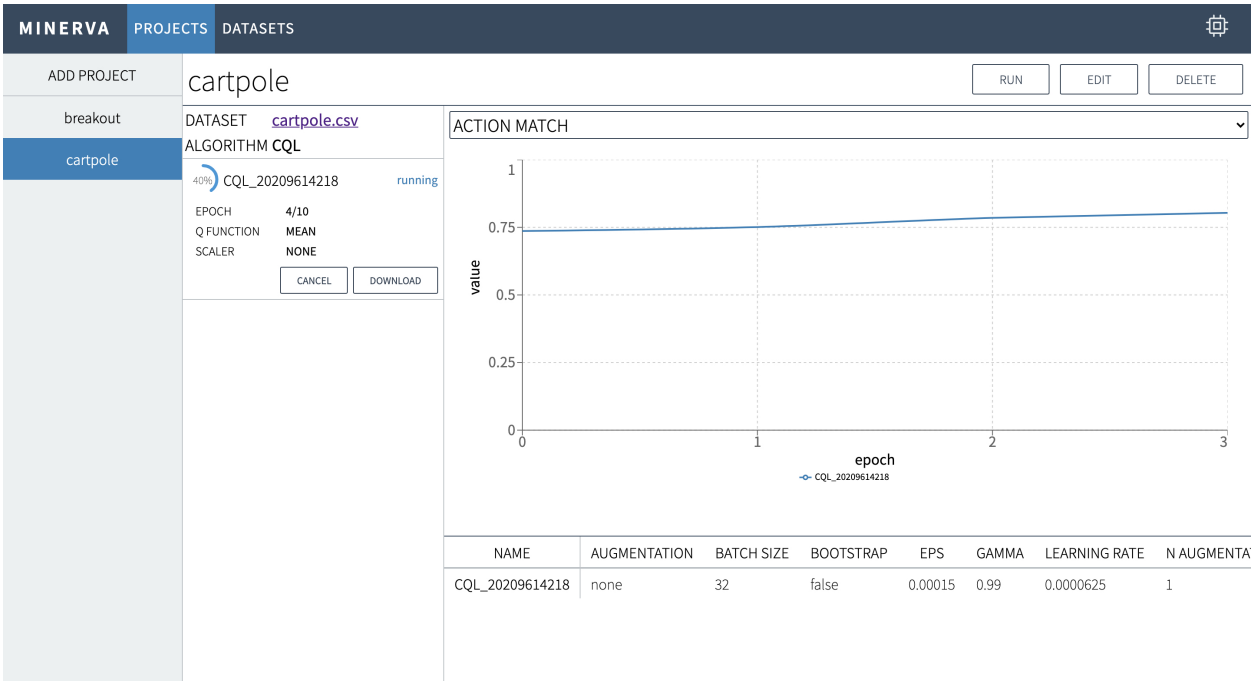
2.5.2 Train with image observation

To train with image observation, you will see different configurations from vector observation projects. The most important option is `N_FRAMES` which controls frame stacking to handle temporal data without recurrent networks.



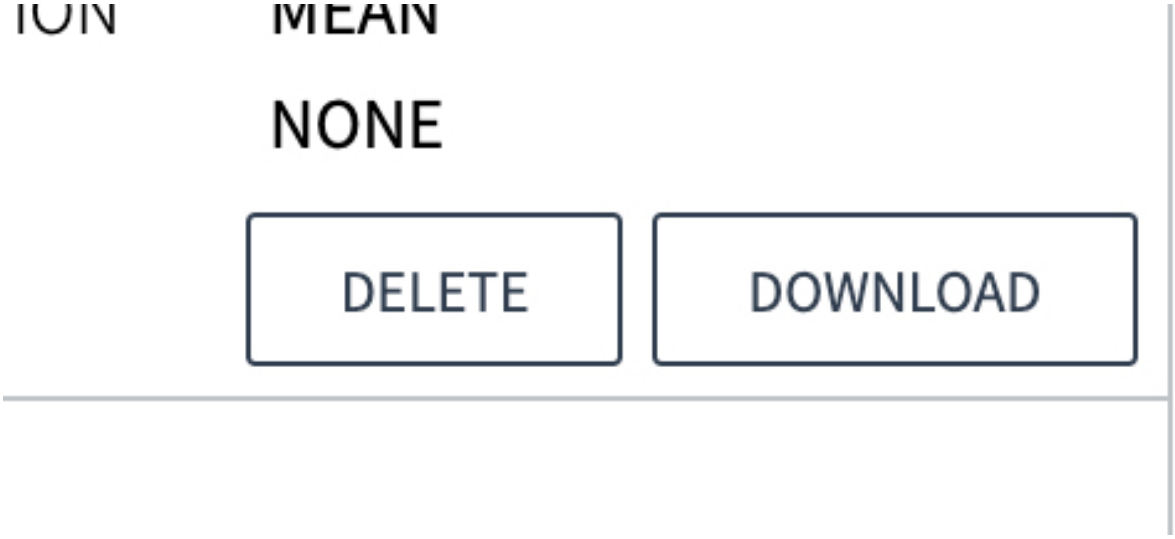
Note: Basically, the `SCALER` option should be set to `PIXEL` when training with image observation.

Once starting training, you will see information about your training. If you need to kill the training process in the middle of training, click `CANCEL` button.



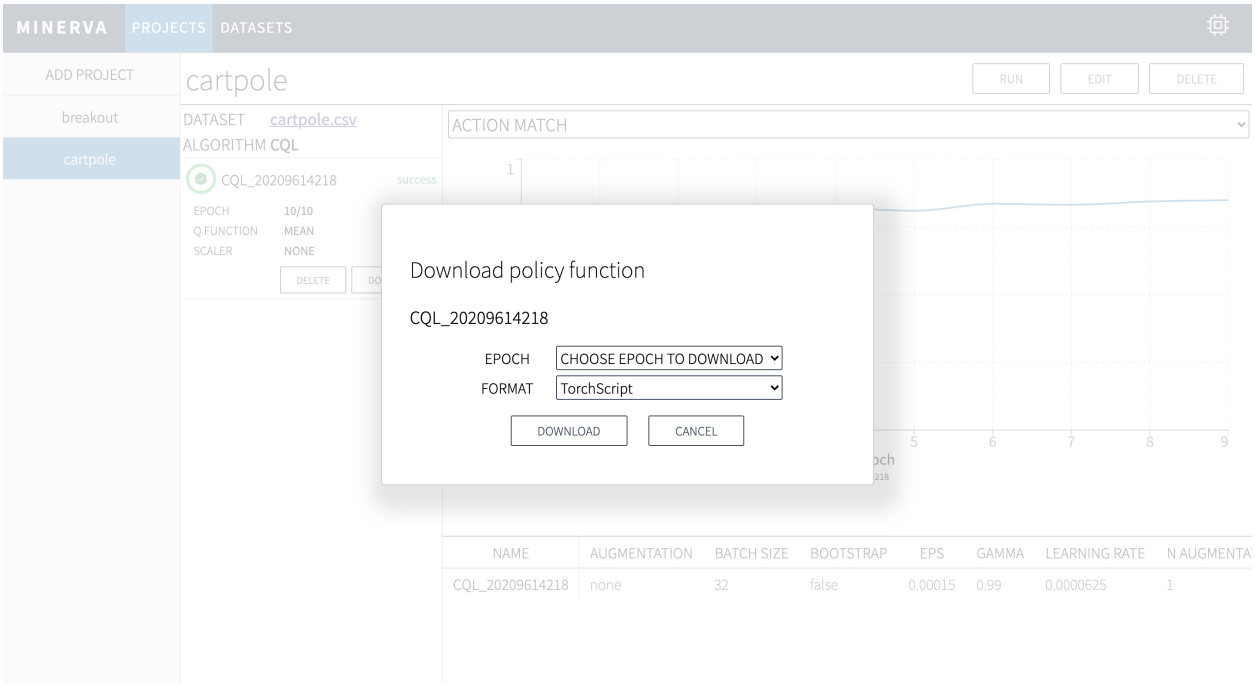
2.6 Export Policy Function

To export the trained policy, click `DOWNLOAD` button.



Then,

1. Choose an epoch to export.
2. Choose a format (e.g. TorchScript and ONNX).
3. Click `DOWNLOAD`.



See how you use the exported policy at *Deploy*.

CHAPTER 3

Tutorials

3.1 CartPole

3.1.1 Download dataset

First of all, download the cartpole dataset as follows:

```
$ wget https://www.dropbox.com/s/vc7fm7qdnu0kh01/cartpole.csv?dl=1 -O cartpole.csv
```

Or access to https://www.dropbox.com/s/vc7fm7qdnu0kh01/cartpole.csv .

3.1.2 Train

Follow instruction from *Upload Dataset* to *Start Training*.

3.1.3 Deploy

Finally, you can download the trained policy as *Export Policy Function*. At this time, you have two options of the model format, TorchScript and ONNX.

TorchScript

You can load the policy in two lines of codes only with PyTorch.

```
import torch

policy = torch.jit.load('policy.pt')
```

It's easy, right?

Then you can write the rest of interaction codes as usual.

```
import gym

env = gym.make('CartPole-v0')

observation = env.reset()

while True:
    # feed observation to the policy
    action = policy(torch.tensor([observation], dtype=torch.float32))

    # take action to get next observation
    observation, _, done, _ = env.step(action[0].numpy())

    # rendering environment
    env.render()

    # break if the episode reaches the termination
    if done:
        break
```

ONNX

In this tutorial, onnxruntime is used to load the model.

```
import onnxruntime as ort

ort_session = ort.InferenceSession('policy.onnx')
```

Basically, ONNX is also easy to load.

Then you can write the rest of interaction codes like above.

```
import gym

env = gym.make('CartPole-v0')

observation = env.reset()

while True:
    # change dtype strictly to float32 and expand its shape
    observation = observation.astype('f4').reshape((1, -1))

    # feed observation to the policy
    action = ort_session.run(None, {'input_0': observation})[0]

    # take action to get next observation
    observation, _, done, _ = env.step(action[0])

    # rendering environment
    env.render()

    # break if the episode reaches the termination
    if done:
        break
```

CHAPTER 4

MINERVA CLI

4.1 run

Run the MINERVA server. To stop, press `Ctrl+C` on the console:

```
$ minerva run
```

- `--host` or `-h`: (optional) set host name (`0.0.0.0` by default).
- `--port` or `-p`: (optional) set port number (`9000` by default).

4.2 clean

Clean all data including the database, the training metrics, and trained parameters:

```
$ minerva clean
```

4.3 upgrade-db

Upgrade database based on the latest schema definitions. This command should be called after version updates:

```
$ minerva upgrade-db
```

4.4 downgrade-db

Downgrade database to the previous revision:

```
$ minerva downgrade-db
```

CHAPTER 5

License

MIT License

Copyright (c) 2020 Takuma Seno

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search