Federated Learning in AI4EOSC

Judith Sáinz-Pardo Díaz (sainzpardo@ifca.unican.es)
IFCA - CSIC

Co-funded by
the European Union

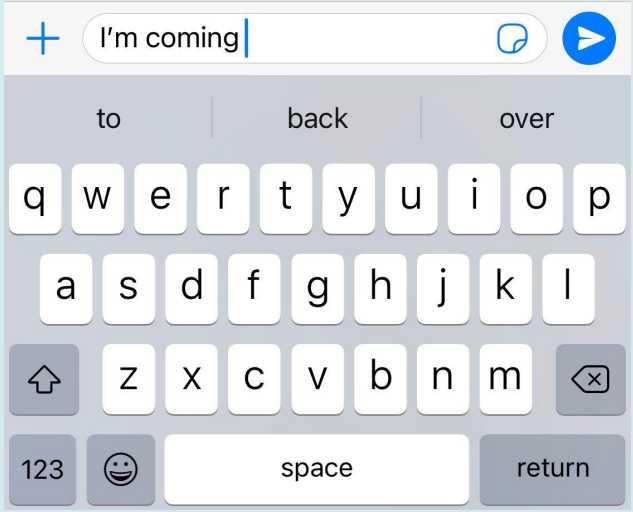
Introduction to Federated Learning

EXAMPLE: Predictive keyboards.

GOAL: use data from different devices to train models that predict the next word.

- *Approach 1*: collect data from all possible devices and train a Machine/Deep Learning model in the cloud. Predictions are returned to each device. In this case the data has to leave the terminal.
  ⟶ CENTRALIZED APPROACH

- *Approach 2*: each device trains in local a model with its own data and makes predictions. The data does not leave the terminal. Less data for training the models.
  ⟶ EDGE COMPUTING



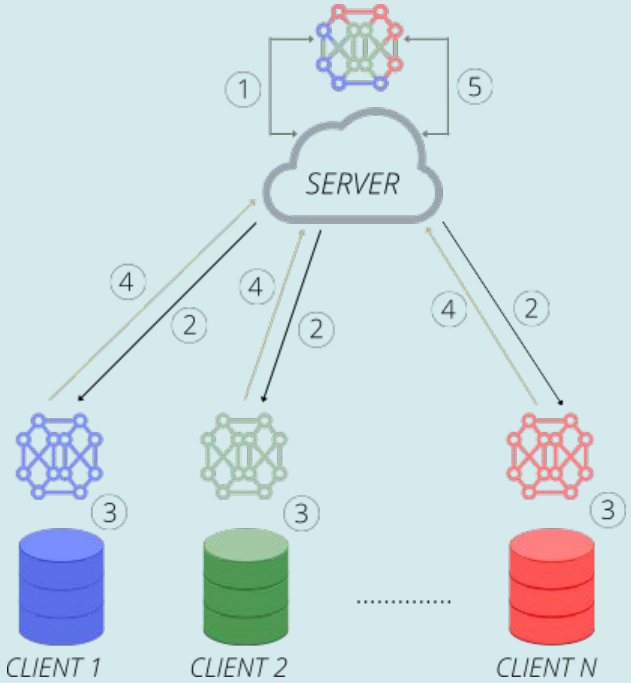
Introduction to Federated Learning

IDEA: Data decentralization.

- Data does not leave the device/center that generates it.
- **SERVER-CLIENT** structure. The server or one client creates the model that will train each client locally with its own data.
- The clients only send to the server the parameters obtained after training the model.
- The server aggregates the weights obtained for the model by each client and updates the initial model.
- It can be seen as a special type of DML.
- Potential clients must be identified, and it must be ensured that they have sufficient and quality data.

Federated Learning: collaborative and decentralized approach to Machine Learning.

Federated Learning (schema)



(1) SERVER: creates the model to be trained locally by each client.

(2) SERVER: transmits the model to the clients.

(3) CLIENT: each of them trains the model with its local data.

(4) CLIENT: each of them sends the local parameters to the server.

(5) SERVER: aggregates the weights of each client using an aggregation operator and updates the model.

Repeats the process from step 2.

Federated Learning vs Centralized approach

- Security and privacy are ensured: data are not shared.
- Reduced communication costs.
- Weights are transferred instead of data.
- Compressing the matrix of numbers that define a model saves bandwidth.
- Greater energy savings.
- Lower computational cost.
- Lower latency.

NOTE: clients can be intermittent (some disappear and new ones enter the training).

Types of Federated Learning

Cross-device FL:
In this Federated Learning approach, the clients are a large number of devices that store sensitive information from different people or entities. *Example: predictive keyboards.*

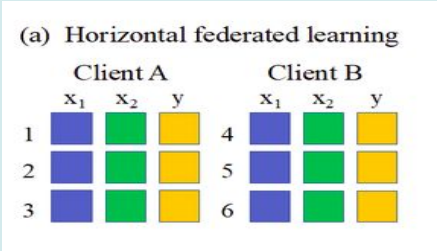
Cross-silo FL:
In this Federated Learning approach, the clients are not devices, but, for example, hospitals, banks, universities, governmental institutions, etc. Likewise, these institutions do not want/cannot share their data with each other or with a central server, so FL is applied. *Example: medical imaging.*

Types of Federated Learning

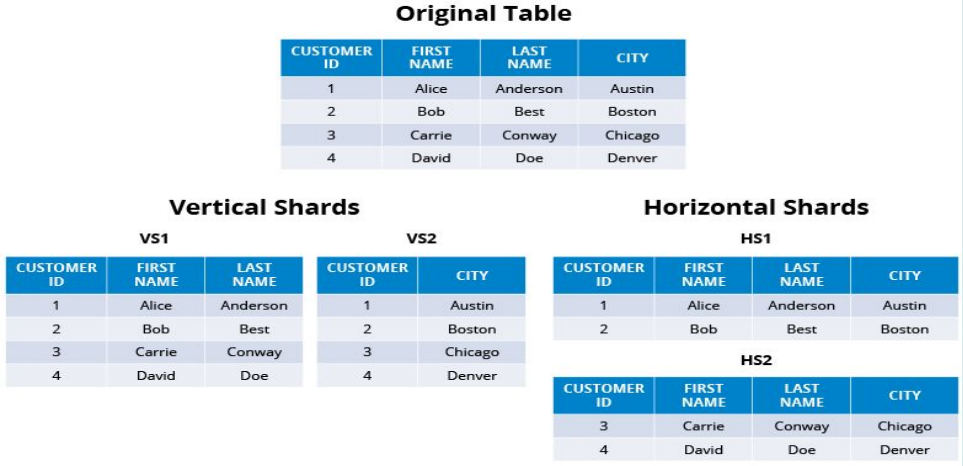
Horizontal FL:
It is the most intuitive and common case. It consists of considering the data of all clients with the same features, for example, in the case of structured data, the data of all the clients will all have the same columns.

Vertical FL:
In this case the different clients have data with different characteristics, but with the same identifier. For example, is the case of several institutions which have data from the same users, but each of them has information about different characteristics. Note that the number of clients will be lower than in the previous case.

Types of Federated Learning



Source: Chen, Shaoqi, et al. "FL-QSAR: a federated learning-based QSAR prototype for collaborative drug discovery." Bioinformatics 36.22-23 (2020): 5492-5498.



Source: https://hazelcast.com/glossary/sharding/

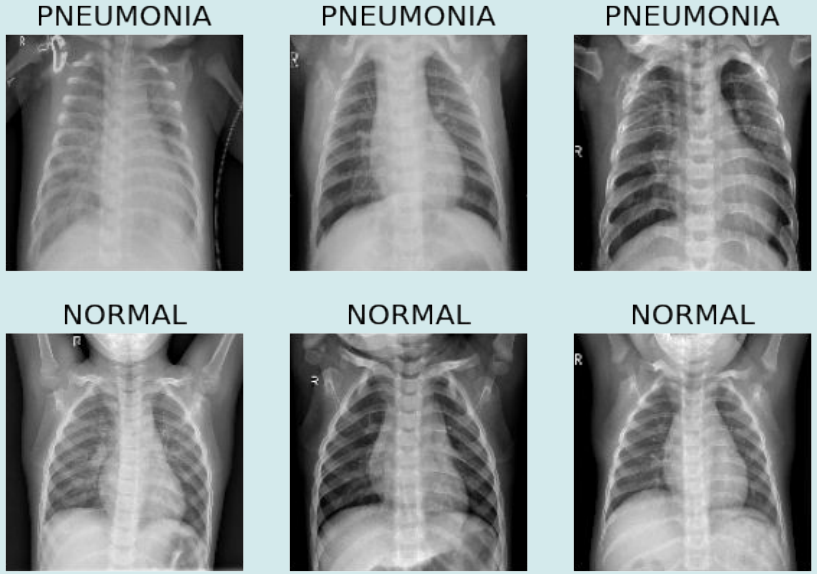
DEMO: Federated Learning in AI4EOSC

GOAL: classify chest X-Ray images according to whether or not the patient has pneumonia.

We divide the initial train data into 3 clients.
Stratified train-test random split:
75% train, 25% test

	Train	Test
Client 1	1050	350
Client 2	1800	600
Client 3	1062	350

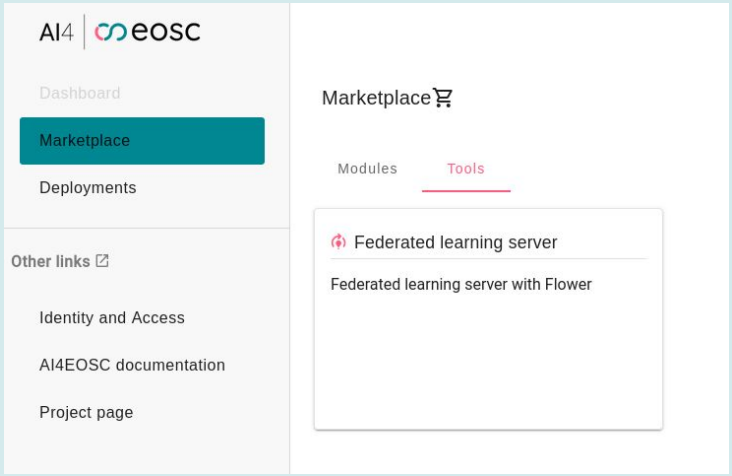
Model: multi-layer convolutional network implemented using keras.

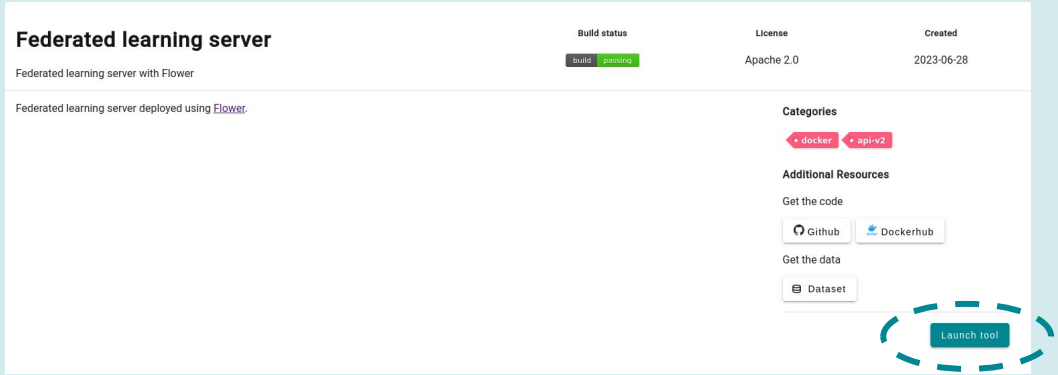


DEMO: Federated Learning in AI4EOSC

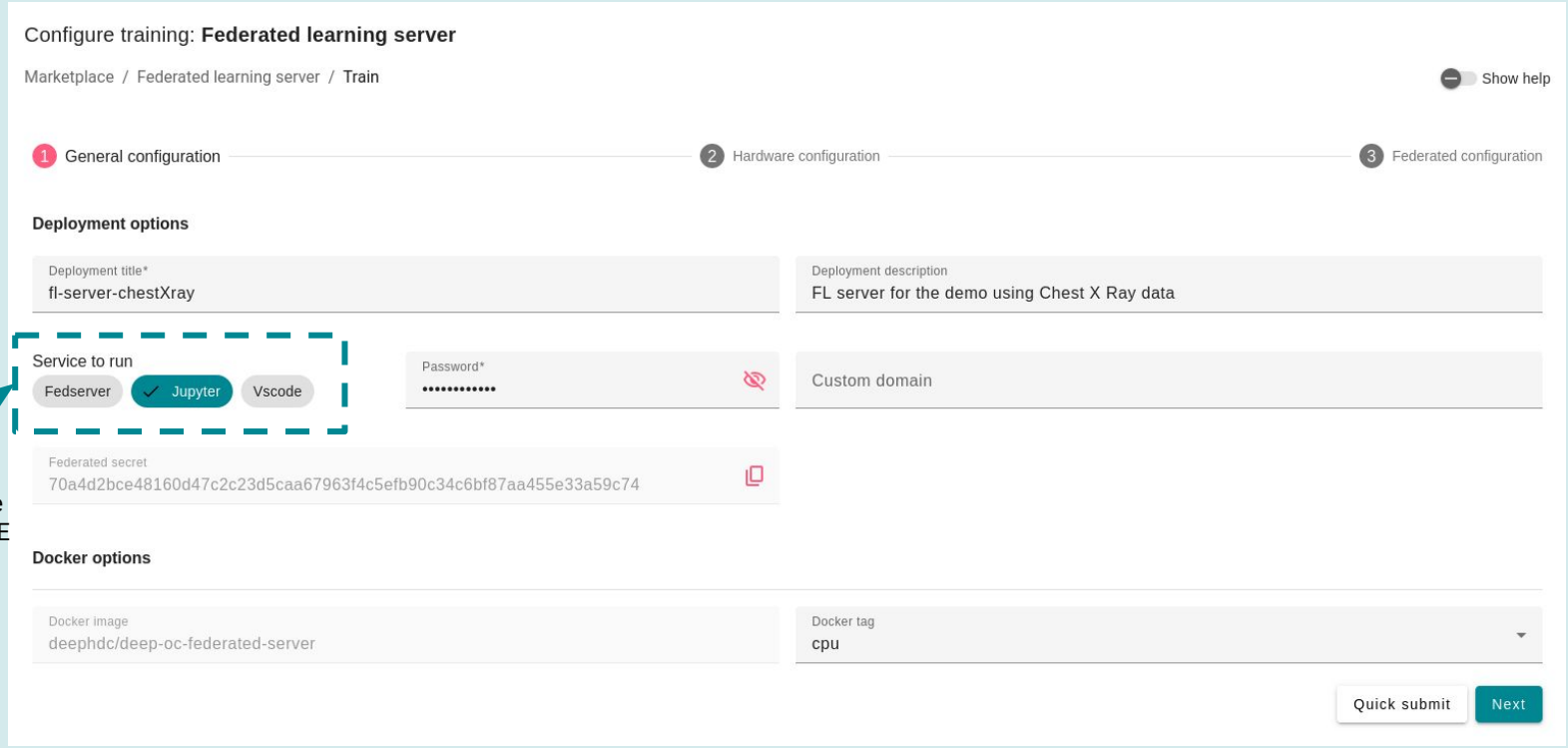
Login in the dashboard (https://dashboard.cloud.ai4eosc.eu/) using EGI check-in.

Create the Federated server:





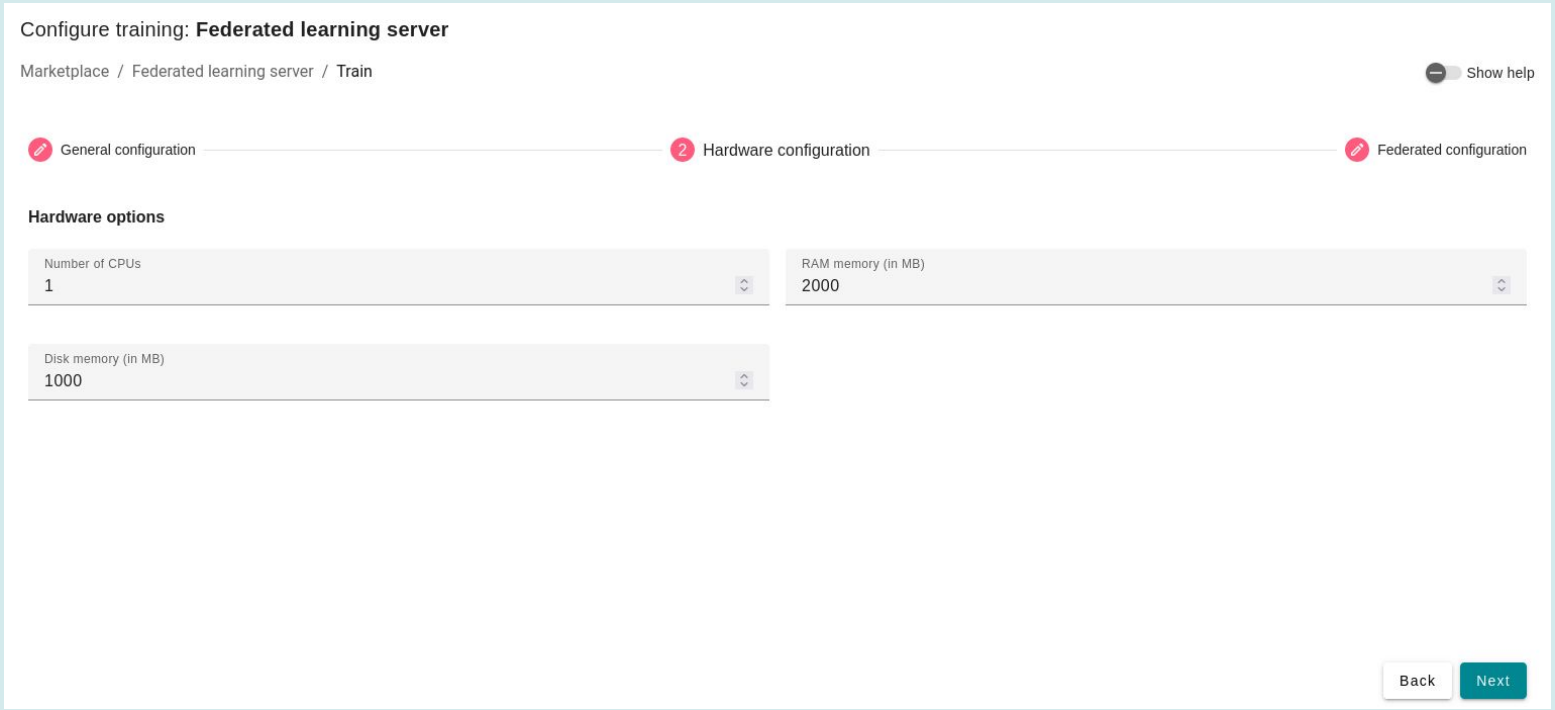
DEMO: Federated Learning in AI4EOSC



Choose your IDE

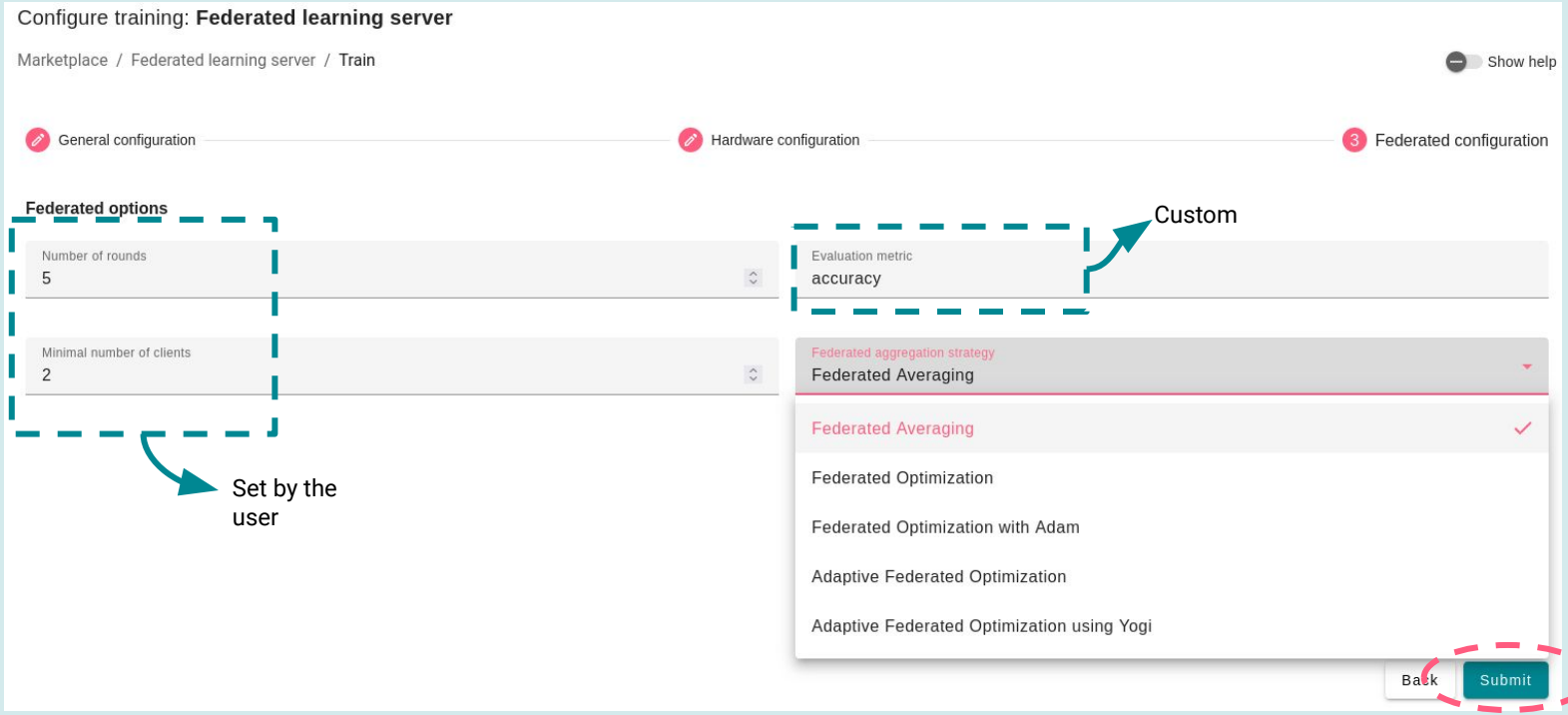
1
General configuration

DEMO: Federated Learning in AI4EOSC



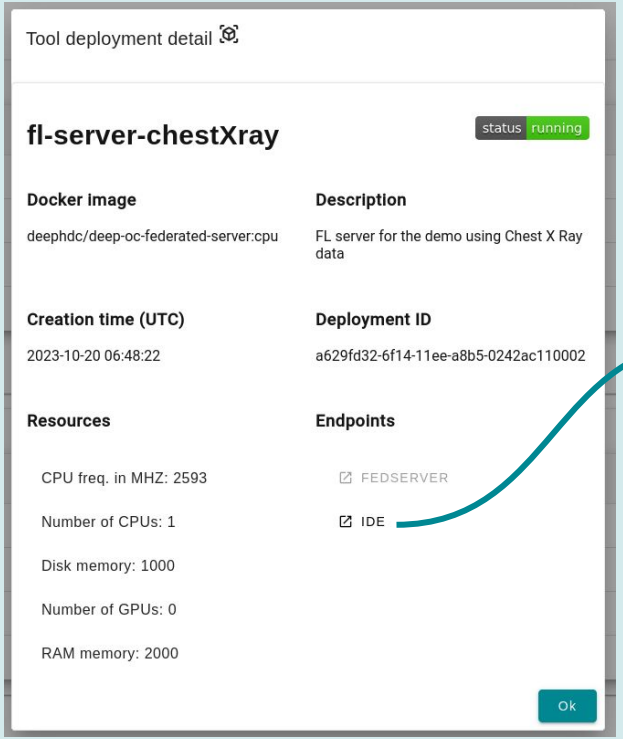
2
Hardware configuration

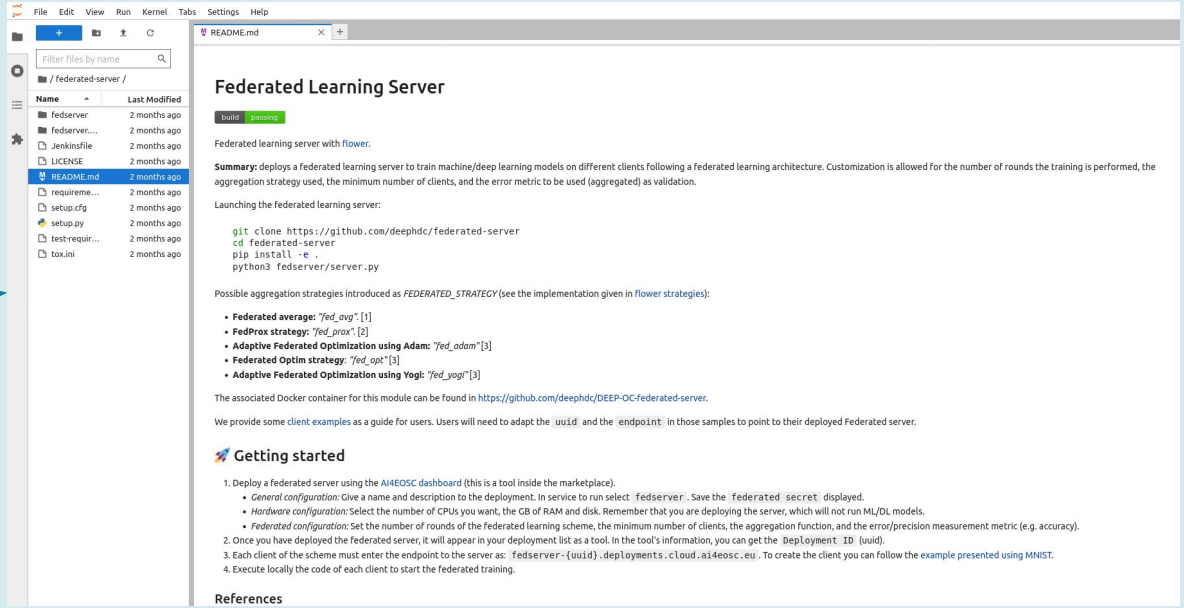
DEMO: Federated Learning in AI4EOSC



3
Federated configuration

DEMO: Federated Learning in AI4EOSC





We are ready for starting the FL server!

DEMO: Federated Learning in AI4EOSC

Create the 3 clients in 3 different machines:

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State
client2_fl	IFCA Ubuntu 20.04 [2023-05-31]	172.16.82.10, 193.146.75.197	cm14-compute-4	key-judith	Active	nova	None	Running
client1_fl	IFCA Ubuntu 20.04 [2023-05-31]	172.16.82.105, 193.146.75.87	cm14-compute-4	key-judith	Active	nova	None	Running

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State
client3_fl	IFCA Ubuntu 20.04 [2023-05-31]	172.16.14.195, 193.146.75.238	cm14-compute-4	key-judith	Active	nova	None	Running

AI4 | eosc

Co-funded by the European Union

DEMO: Federated Learning in AI4EOSC

Client 1

Create the 3 clients in 3 different machines:

```
judith@Latitude:~$ ssh ubuntu@193.146.75.87
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-149-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct 20 07:11:30 UTC 2023

  System load:  0.05               Processes:             133
  Usage of /:   32.7% of 19.20GB   Users logged in:       0
  Memory usage: 5%                 IPv4 address for ens3: 172.16.82.105
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Fri Oct 20 07:10:58 2023 from 193.144.210.100
ubuntu@client1-fl:~$ cd client1_chestxray/
ubuntu@client1-fl:~/client1_chestxray$ ls
client1.py  x_client1.npy  y_client1.npy
ubuntu@client1-fl:~/client1_chestxray$
```

```python
# Load and process data:
x_client1 = np.fromfile('x_client1.npy').reshape((1400, 150, 150))
y_client1 = np.load('y_client1.npy')
x_client1 = np.reshape(x_client1, (1400, 150, 150, 1))

x_train, x_test, y_train, y_test = train_test_split(x_client1, y_client1,
                                                    test_size=0.25, random_state=42, stratify = y_client1)
```

```python
# Flower client
class Client1(fl.client.NumPyClient):
    def get_parameters(self, config):
        return model.get_weights()

    def fit(self, parameters, config):
        model.set_weights(parameters)
        model.fit(x_train, y_train, epochs=5, batch_size=16)
        return model.get_weights(), len(x_train), {}

    def evaluate(self, parameters, config):
        model.set_weights(parameters)
        loss, accuracy = model.evaluate(x_test, y_test)
        return loss, len(x_test), {"accuracy": accuracy}
```

```python
# Start -> connecting with the server
uuid = 'a68d6250-6f14-11ee-992e-0242ac110002'
end_point = f"fedserver-{uuid}.deployments.cloud.ai4eosc.eu"
fl.client.start_numpy_client(
    server_address=f"{end_point}:443",
    client=Client1(),
    root_certificates=Path(certifi.where()).read_bytes()
)
```

FL server UUID

DEMO: Federated Learning in AI4EOSC

Client 2

Create the 3 clients in 3 different machines:

```
judith@Latitude:~$ ssh ubuntu@193.146.75.197
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-149-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct 20 07:27:59 UTC 2023

  System load:  0.0               Processes:             131
  Usage of /:   28.7% of 19.20GB  Users logged in:       0
  Memory usage: 5%                IPv4 address for ens3: 172.16.82.10
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Fri Oct 20 07:27:42 2023 from 193.144.210.100
ubuntu@client2-fl:~$ cd client2_chestxray/
ubuntu@client2-fl:~/client2_chestxray$ ls
client2.py  x_client2.npy  y_client2.npy
ubuntu@client2-fl:~/client2_chestxray$
```

```python
# Load and process data:
x_client2 = np.fromfile('x_client2.npy').reshape((2400, 150, 150))
y_client2 = np.load('y_client2.npy')
x_client2 = np.reshape(x_client2, (2400, 150, 150, 1))

x_train, x_test, y_train, y_test = train_test_split(x_client2, y_client2,
                                                    test_size=0.25, random_state=42, stratify = y_client2)
```

```python
# Flower client
class Client2(fl.client.NumPyClient):
    def get_parameters(self, config):
        return model.get_weights()

    def fit(self, parameters, config):
        model.set_weights(parameters)
        model.fit(x_train, y_train, epochs=5, batch_size=16)
        return model.get_weights(), len(x_train), {}

    def evaluate(self, parameters, config):
        model.set_weights(parameters)
        loss, accuracy = model.evaluate(x_test, y_test)
        return loss, len(x_test), {"accuracy": accuracy}
```

```python
# Start -> connecting with the server
uuid = 'a68d6250-6f14-11ee-992e-0242ac110002'
end_point = f"fedserver-{uuid}.deployments.cloud.ai4eosc.eu"
fl.client.start_numpy_client(
    server_address=f"{end_point}:443",
    client=Client2(),
    root_certificates=Path(certifi.where()).read_bytes()
)
```

FL server UUID

DEMO: Federated Learning in AI4EOSC

Client 3

Create the 3 clients in 3 different machines:

```
judith@Latitude:~$ ssh ubuntu@193.146.75.238
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-149-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct 20 07:01:36 UTC 2023

  System load:  0.0               Processes:             134
  Usage of /:   29.3% of 19.20GB  Users logged in:       0
  Memory usage: 5%                IPv4 address for ens3: 172.16.14.195
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Fri Oct 20 07:01:04 2023 from 193.144.210.100
ubuntu@client3-fl:~$ cd client3_chestxray/
ubuntu@client3-fl:~/client3_chestxray$ ls
client3.py  x_client3.npy  y_client3.npy
ubuntu@client3-fl:~/client3_chestxray$
```

```python
# Load and process data:
x_client3 = np.fromfile('x_client3.npy').reshape((1416, 150, 150))
y_client3 = np.load('y_client3.npy')
x_client3 = np.reshape(x_client3, (1416, 150, 150, 1))

x_train, x_test, y_train, y_test = train_test_split(x_client3, y_client3,
                                                    test_size=0.25, random_state=42, stratify = y_client3)
```

```python
# Flower client
class Client3(fl.client.NumPyClient):
    def get_parameters(self, config):
        return model.get_weights()

    def fit(self, parameters, config):
        model.set_weights(parameters)
        model.fit(x_train, y_train, epochs=5, batch_size=16)
        return model.get_weights(), len(x_train), {}

    def evaluate(self, parameters, config):
        model.set_weights(parameters)
        loss, accuracy = model.evaluate(x_test, y_test)
        return loss, len(x_test), {"accuracy": accuracy}
```

```python
# Start -> connecting with the server
uuid = 'a68d6250-6f14-11ee-992e-0242ac110002'
end_point = f"fedserver-{uuid}.deployments.cloud.ai4eosc.eu"
fl.client.start_numpy_client(
    server_address=f"{end_point}:443",
    client=Client3(),
    root_certificates=Path(certifi.where()).read_bytes()
)
```

FL server UUID

DEMO: Federated Learning in AI4EOSC

Model to be trained (same for the three clients)

```
# Model to be trained:
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(128 , activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = 'adam' , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model.summary()
```

DEMO: Federated Learning in AI4EOSC

After performing the federated training, each client use it to predict in the test split.
We show some error metrics.

Example with the first client:

```
# Start -> connecting with the server
uuid = 'a68d6250-6f14-11ee-992e-0242ac110002'
end_point = f"fedserver-{uuid}.deployments.cloud.ai4eosc.eu"
fl.client.start_numpy_client(
    server_address=f"{end_point}:443",
    client=Client1(),
    root_certificates=Path(certifi.where()).read_bytes()
)


score = model.evaluate(x_test, y_test)
pred = model.predict(x_test)
fpr, tpr, _ = metrics.roc_curve(y_test, pred)
auc = metrics.auc(fpr, tpr)
print(f'CLIENT 1: Test loss: {score[0]} / Test accuracy: {score[1]} / Test AUC: {auc}')

plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='AUC = {:.3f}'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title(f'ROC curve. Client 1')
plt.legend(loc='best')
plt.savefig(f"roc_client1.png")
```

DEMO: Federated Learning in AI4EOSC

PERFORM THE FEDERATED TRAINING:

1. Start the federated server:

```
root@c59b3d179e32:/srv# cd federated-server/fedserver
root@c59b3d179e32:/srv/federated-server/fedserver# python3 server.py
INFO flwr 2023-10-20 07:48:47,668 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=10, round_timeout=None)
INFO flwr 2023-10-20 07:48:47,680 | app.py:175 | Flower ECE: gRPC server running (10 rounds), SSL is disabled
INFO flwr 2023-10-20 07:48:47,680 | server.py:89 | Initializing global parameters
INFO flwr 2023-10-20 07:48:47,680 | server.py:276 | Requesting initial parameters from one random client
```

DEMO: Federated Learning in AI4EOSC

PERFORM THE FEDERATED TRAINING:

1. Start the federated server.
2. Start the clients:
   Example with client 1:

```
(.venv) ubuntu@client1-fl:~/client1_chestxray$ source .venv/bin/activate
(.venv) ubuntu@client1-fl:~/client1_chestxray$ python3 client1.py
```

```
INFO flwr 2023-10-20 07:49:02,970 | grpc.py:46 | Opened secure gRPC connection using certificates
DEBUG flwr 2023-10-20 07:49:02,974 | connection.py:39 | ChannelConnectivity.IDLE
DEBUG flwr 2023-10-20 07:49:02,974 | connection.py:39 | ChannelConnectivity.CONNECTING
DEBUG flwr 2023-10-20 07:49:03,250 | connection.py:39 | ChannelConnectivity.READY
```

Client side

```
INFO flwr 2023-10-20 07:48:47,668 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=10, round_timeout=None)
INFO flwr 2023-10-20 07:48:47,680 | app.py:175 | Flower ECE: gRPC server running (10 rounds), SSL is disabled
INFO flwr 2023-10-20 07:48:47,680 | server.py:89 | Initializing global parameters
INFO flwr 2023-10-20 07:48:47,680 | server.py:276 | Requesting initial parameters from one random client
INFO flwr 2023-10-20 07:49:03,357 | server.py:280 | Received initial parameters from one random client
INFO flwr 2023-10-20 07:49:03,357 | server.py:91 | Evaluating initial parameters
INFO flwr 2023-10-20 07:49:03,357 | server.py:104 | FL starting
```

Server side

DEMO: Federated Learning in AI4EOSC

PERFORM THE FEDERATED TRAINING:

1. Start the federated server.
2. Start the clients:
   - Start client 1
   - Start client 2

Server side (round 4)

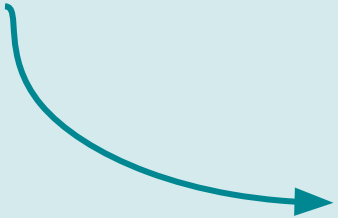
```
INFO flwr 2023-10-20 07:48:47,668 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=10, round_timeout=None)
INFO flwr 2023-10-20 07:48:47,680 | app.py:175 | Flower ECE: gRPC server running (10 rounds), SSL is disabled
INFO flwr 2023-10-20 07:48:47,680 | server.py:89 | Initializing global parameters
INFO flwr 2023-10-20 07:48:47,680 | server.py:276 | Requesting initial parameters from one random client
INFO flwr 2023-10-20 07:49:03,357 | server.py:280 | Received initial parameters from one random client
INFO flwr 2023-10-20 07:49:03,357 | server.py:91 | Evaluating initial parameters
INFO flwr 2023-10-20 07:49:03,357 | server.py:104 | FL starting
DEBUG flwr 2023-10-20 07:51:32,772 | server.py:222 | fit_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:52:26,227 | server.py:236 | fit_round 1 received 2 results and 0 failures
WARNING flwr 2023-10-20 07:52:26,275 | fedavg.py:242 | No fit_metrics_aggregation_fn provided
DEBUG flwr 2023-10-20 07:52:26,276 | server.py:173 | evaluate_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:52:27,527 | server.py:187 | evaluate_round 1 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:52:27,527 | server.py:222 | fit_round 2: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:53:20,754 | server.py:236 | fit_round 2 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:53:20,783 | server.py:173 | evaluate_round 2: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:53:21,828 | server.py:187 | evaluate_round 2 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:53:21,828 | server.py:222 | fit_round 3: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:54:13,985 | server.py:236 | fit_round 3 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:54:14,019 | server.py:173 | evaluate_round 3: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:54:14,996 | server.py:187 | evaluate_round 3 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:54:14,996 | server.py:222 | fit_round 4: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:55:06,559 | server.py:236 | fit_round 4 received 2 results and 0 failures
```

DEMO: Federated Learning in AI4EOSC

PERFORM THE FEDERATED TRAINING:

1. Start the federated server.
2. Start the clients:
   - Start client 1
   - Start client 2
   - Start client 3 later (e.g. after 4 rounds)

Server side



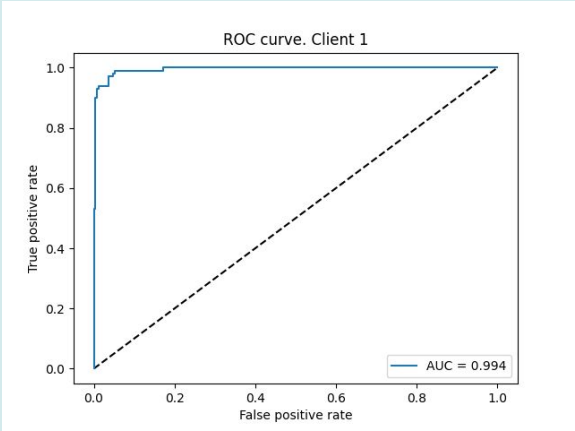
```
DEBUG flwr 2023-10-20 07:54:14,996 | server.py:222 | fit_round 4: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-10-20 07:55:06,559 | server.py:236 | fit_round 4 received 2 results and 0 failures
DEBUG flwr 2023-10-20 07:55:06,586 | server.py:173 | evaluate_round 4: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-10-20 07:55:08,357 | server.py:187 | evaluate_round 4 received 3 results and 0 failures
DEBUG flwr 2023-10-20 07:55:08,358 | server.py:222 | fit_round 5: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-10-20 07:56:00,082 | server.py:236 | fit_round 5 received 3 results and 0 failures
DEBUG flwr 2023-10-20 07:56:00,130 | server.py:173 | evaluate_round 5: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-10-20 07:56:01,194 | server.py:187 | evaluate_round 5 received 3 results and 0 failures
DEBUG flwr 2023-10-20 07:56:01,194 | server.py:222 | fit_round 6: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-10-20 07:56:52,889 | server.py:236 | fit_round 6 received 3 results and 0 failures
DEBUG flwr 2023-10-20 07:56:52,934 | server.py:173 | evaluate_round 6: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-10-20 07:56:53,950 | server.py:187 | evaluate_round 6 received 3 results and 0 failures
DEBUG flwr 2023-10-20 07:56:53,950 | server.py:222 | fit_round 7: strategy sampled 3 clients (out of 3)
```

DEMO: Federated Learning in AI4EOSC

Results for each client after completing the federated training:

CLIENT 1: Test loss: 0.17493936419487 / Test accuracy: 0.9657142758369446 / Test AUC: 0.99436
CLIENT 2: Test loss: 0.06348654627799988 / Test accuracy: 0.9800000190734863 / Test AUC: 0.99896
CLIENT 3: Test loss: 0.5466659665107727 / Test accuracy: 0.9067796468734741 / Test AUC: 0.9860646034162015

Example ROC for client 1:



Conclusions

- An example of federated learning has been carried out using the AI4EOSC platform.
- *Jupyter notebook* has been used as IDE, but *fedserver* or *visual studio code* could be used.
- Three clients have been considered (from a *medical imaging use case*), the third of them intermittent, since it enters the training later than the other two.
- These three clients were distributed on three different cloud machines.
- Robust results in terms of accuracy and AUC were obtained for the test sets of the three clients.
- **Work under development:**
  - Use the *federated secret* (token) for authentication. Already deployed using call credentials in gRPC (https://github.com/AI4EOSC/flower/tree/develop). Discussion already opened in flower (https://github.com/orgs/adap/discussions/1487)
  - Uses cases including differential privacy → medical imaging use case.
  - AI4EOSC use cases: working with data from UC1.

NOTE: If gRPC server is running behind a load balancer (as in our case, Traefik), clients may not be able to connect. Flower is using the peer() method from grpc.ServicerContext in order to identify unique flower clients. However, in some situations (like when running the gRPC server behind a load balancer or proxy) different clients can have the same peer identifier (i.e. the same IP:PORT), as HTTP/2 connections are multiplexed.
We have opened an issue and implemented our own version of the library changing this issue:https://github.com/AI4EOSC/flower/commit/b215d9f3cce1ad8806e296db4fe105a8b7f5c6c9

AI4 | eosc

Co-funded by the European Union

AI4EOSC

ai4eosc-po@listas.csic.es

ai4eosc.eu

Thank you for your attention!

Judith Sáinz-Pardo Díaz - sainzpardo@ifca.unican.es