

# ADDING ALGORITHMS TO BEAT

## Setup:

1. Log into the BEAT platform (<https://www.beat-eu.org/platform/>)
2. Go to User Resources > Algorithms (see <https://www.idiap.ch/software/beat/docs/beat/docs/master/beat/beat.web/doc/user/algorithms/guide.html> for more information)
3. Find the algorithm “jmcgrat3/LivDet-Iris-2020-Base/1” and click the purple fork icon to create a new algorithm from this algorithm. This will give you a base algorithm to work with.

Fork / algorithms / jmcgrat3 / LivDet-Iris-2020-Base / 1 Save Cancel

Name:

The name for this algorithm (space-like characters will be automatically replaced by dashes)

Analyzer  Splittable

Type:  Sequential  Autonomous  
The API used by this algorithm

Language:  Python  C++  
The language you'll use to write this algorithm. Note that this cannot be changed later!

Endpoints ^

Algorithms must declare at least one **input** and one **output**, organized in **groups**. Groups are used by the platform to indicate which inputs and outputs are synchronized together. Outputs can only be declared in the **first group**. The inputs in the **last group** can be declared as not synchronized together (to avoid the creation of several groups with only one input each). The first group is automatically synchronized with the channel defined by the block in which the algorithm is deployed.

Inputs / Outputs: +

Inputs:  system/array\_2d\_uint8/1 +

Outputs:  system/int64/1 +

4. Add any parameters your algorithm will need in the parameters section

Parameters ^

Parameters allow users to change the configuration of an algorithm when scheduling an experiment

Name	Type	Range	Choices	Default	Description
<input type="text" value="score_to_output"/>	<input type="text" value="float64"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="50"/>	<input type="text" value="the score to output (sample param)"/>

+

## Development:

### Python

1. The editor window should already be set up with the necessary code.

Source code:

```
1 class Algorithm:
2     def setup(self, parameters):
3         # perform some setup using the parameters
4         # here, we set the score to output
5         self.score_to_output = parameters['score_to_output']
6         return True
7
8     def process(self, inputs, data_loaders, outputs):
9         # read the image from the input (will be np array)
10        image = inputs['image'].data.value
11        # perform some processing
12        output = self.score_to_output
13        # write to outputs
14        outputs['score'].write({'value':output})
15        return True
16
```

2. Import an additional modules your algorithm will need (make sure they are included in the environment - [http://www.iris2020.livdet.org/wp-content/uploads/2020/06/environment\\_beat.txt](http://www.iris2020.livdet.org/wp-content/uploads/2020/06/environment_beat.txt))
3. Any parameters your algorithm needs (that you added in setup step 4 above) can be loaded in the *setup* portion of the algorithm. See the examples (<https://www.idiap.ch/software/beat/docs/beat/docs/master/beat/beat.web/doc/user/algorithms/guide.html>) for more details
4. The inputs and outputs of the *process* portion of the algorithm should remain the same.
5. If your algorithm relies on external models, they will need to be converted to a text format and added to the *Algorithm* class

```
model = b'\x80\x03csklearn.discriminant_analysis\nLinearDiscriminantAnalysis\nq\x00)\x81q\x01}q\x02(X\
lda = pickle.loads(model)
lda
```

## C++

1. Follow the instructions for setting up the environment to develop in C++ (<https://www.idiap.ch/software/beat/docs/beat/docs/master/beat/beat.web/doc/user/algorithms/guide.html#implementing-an-algorithm-in-c>)
  - a. The docker image can be found at <https://hub.docker.com/r/beatenv/beat.env.cxxdev>
2. Change the algorithm you created in setup to be a C++ algorithm and save it.
3. Continue following the instructions from step 1 (pull your algorithm into the environment and run `generate_cxx.py` to create the necessary files)
4. Follow the examples to retrieve your parameters and inputs and to set your outputs
  - a. <https://www.idiap.ch/software/beat/docs/beat/docs/master/algorithms.html#preparation-of-an-algorithm>
  - b. Boost data types are used in C++
5. Make your `.so` file
  - a. Make sure to update the `CMake` file to correctly link any dependencies you may need
6. Upload the file to your algorithm on the BEAT website

Type:  Sequential  Autonomous  
The API used by this algorithm

Endpoints ▼

Parameters ▼

i This algorithm is implemented in C++, compiled as a shared library.

Shared library: + Select file...  
The compiled shared library file implementing your algorithm

**If you have issues when implementing your algorithm for BEAT, please ask us at [jmcgrat3@alumni.nd.edu](mailto:jmcgrat3@alumni.nd.edu)**