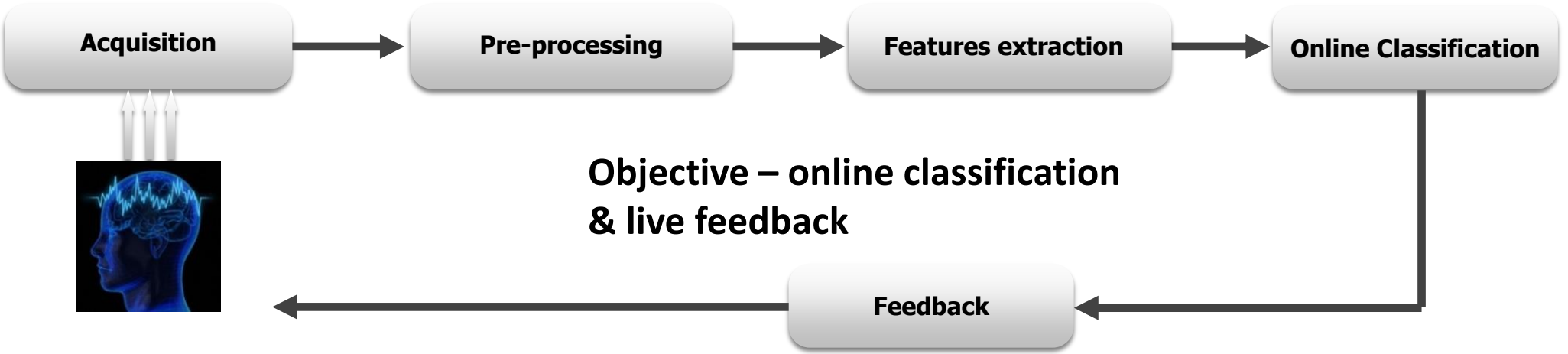
December 7th 2022

HappyFeat: an interactive and efficient BCI framework for clinical applications

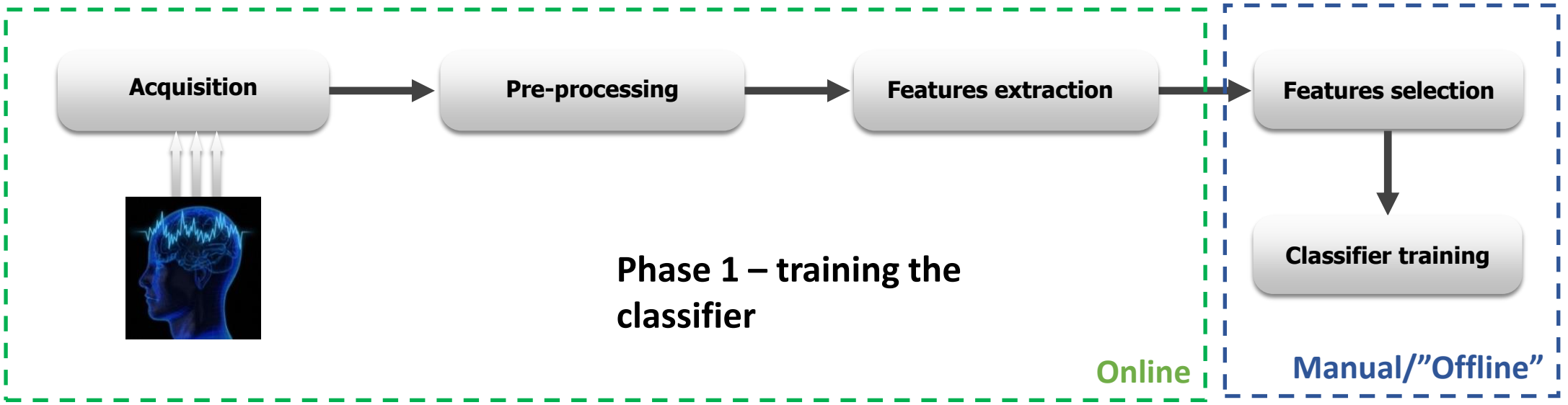
Arthur Desbois, Inria Paris

ARAMIS team, Paris Brain Institute

What is “BCI”?

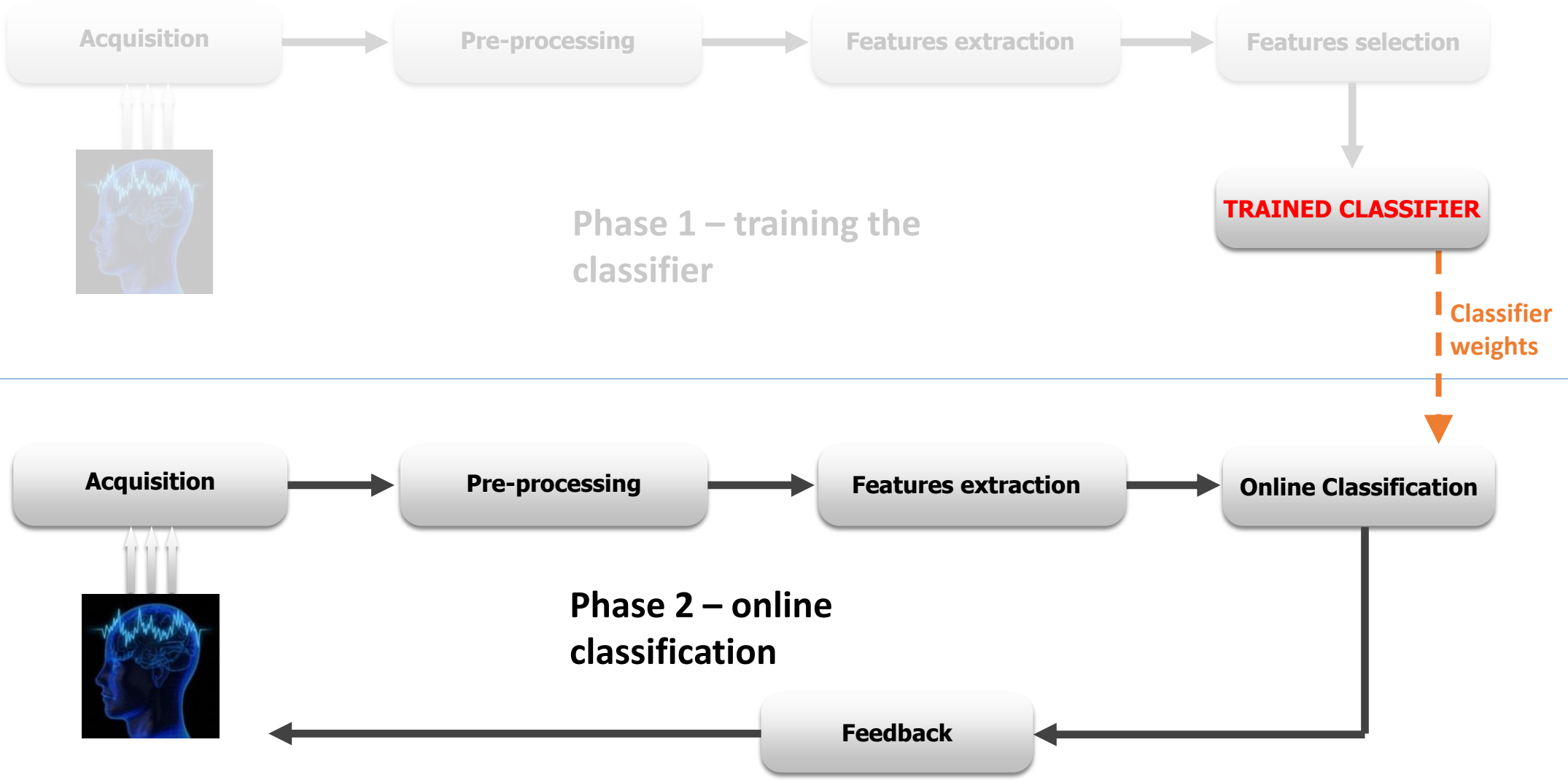

Acquisition
Pre-processing
Features extraction
Online Classification
Objective – online classification
& live feedback
Feedback

What is “BCI”?


Acquisition
Pre-processing
Features extraction
Features selection
Classifier training
Phase 1 – training the classifier
Online
Manual/”Offline”

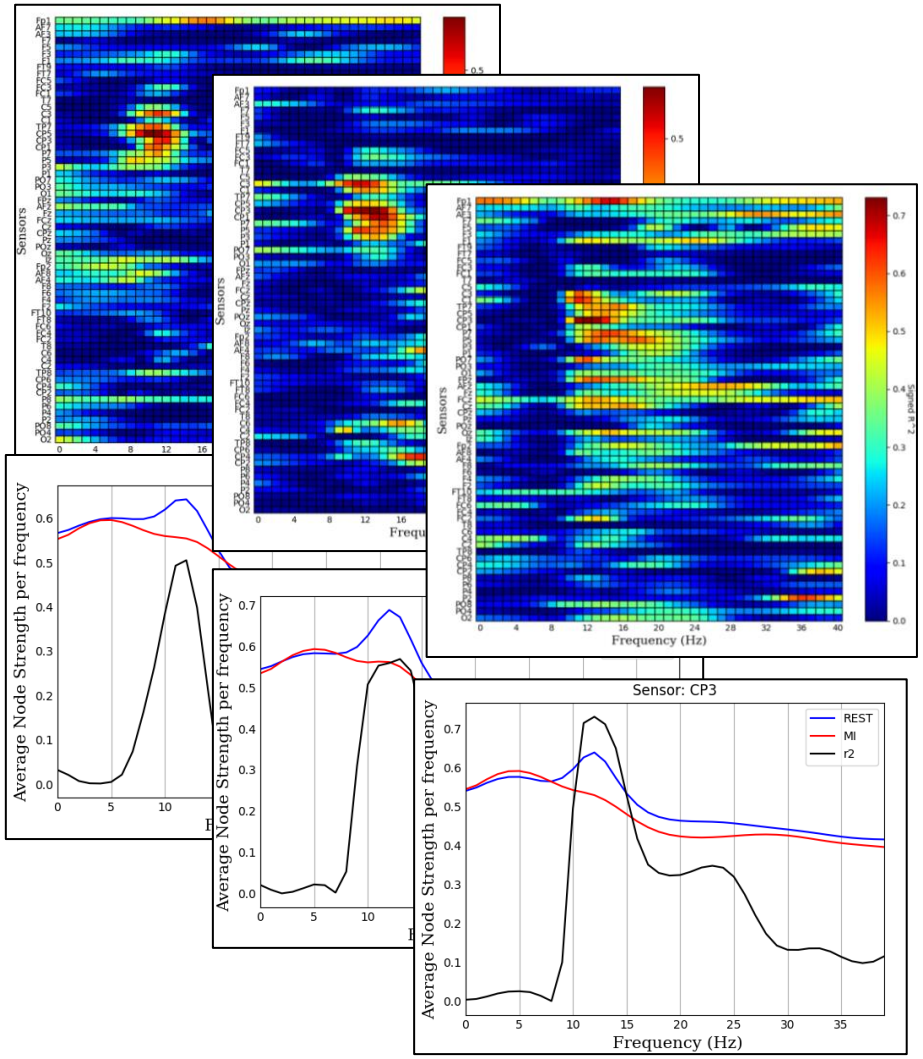
Acquisition
Pre-processing
Features extraction
Online Classification
Objective – online classification & live feedback
Feedback

What is “BCI”?


Acquisition
Pre-processing
Features extraction
Features selection
TRAINED CLASSIFIER
Phase 1 – training the classifier
Classifier weights
Acquisition
Pre-processing
Features extraction
Online Classification
Phase 2 – online classification
Feedback

BCI in clinical settings - Feature selection

- **Features of interest (FOI)**
  - Selecting adequate FOIs is a **crucial step for BCI performance**.
  - After EEG signals acquisition, an **analysis phase** is needed to select best FOIs.
    - ➔ Scientific softwares (i.e., MATLAB)
    - ➔ Manual step, expertise needed
  - If this analysis phase is too long, a lot can change in the meantime:
    - EEG sensors impedances
    - Subject's brain behaviors
    - Subject's attention & motivation



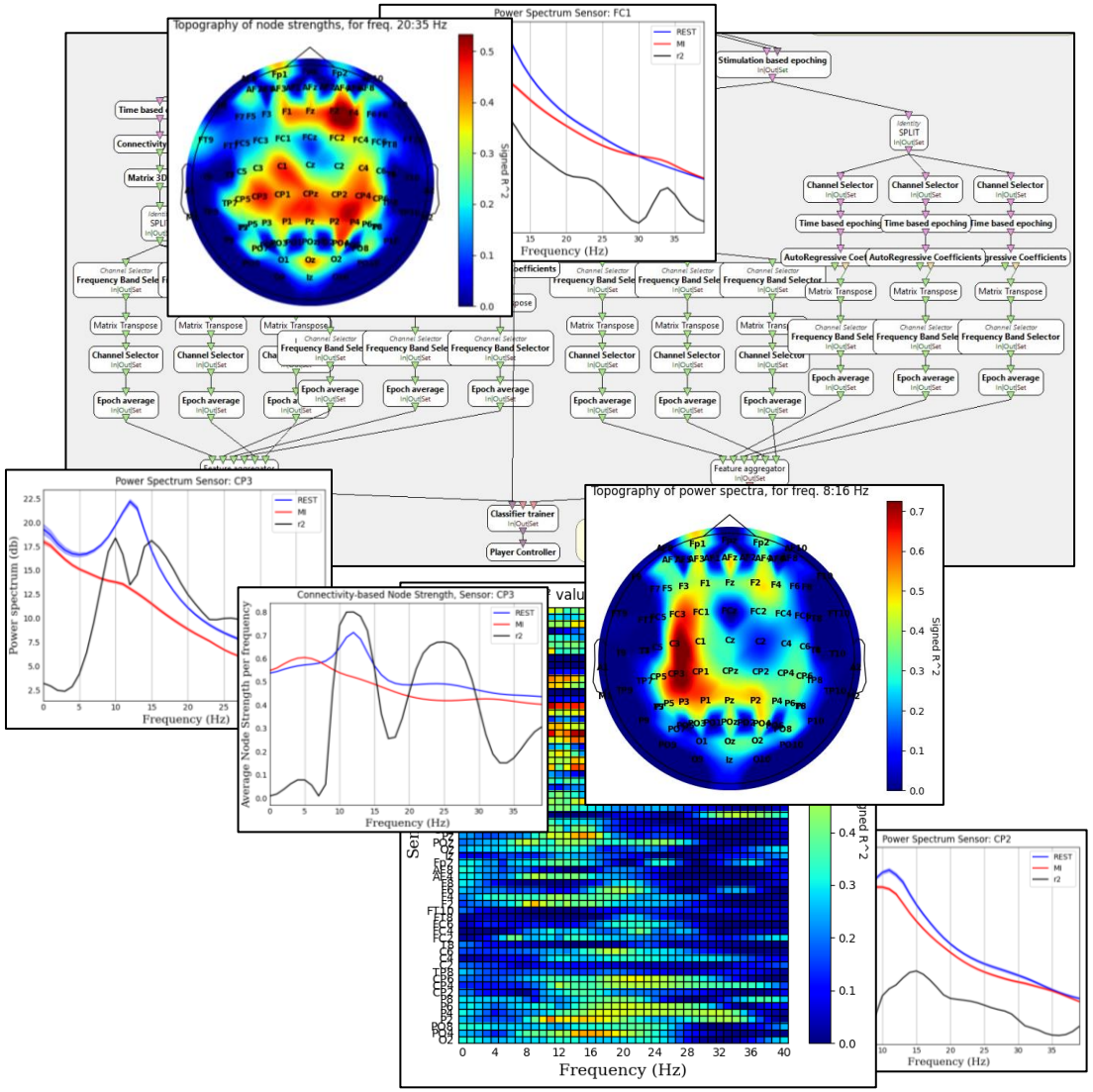
➔ Signal characteristics might be very different between Acq/Training phase and Online phase…

BCI in clinical settings - Feature selection

The “analysis phase” involves many manipulations:

- Setting up “**feature extraction**” scenarios in OpenViBE…
- Finding FOIs through **visualization**…
- Setting up & running **training** scenarios in OpenViBE…
- … and maybe **going again** through those steps multiple times until “correct” features have been found, or to account for inter-run variability

➔ **Tedious, error-prone, hard to achieve in a limited time**



HappyFeat - Main Concepts

Python-based framework for facilitating MI pipelines

Main focus:

making Feature Selection

+ Classif. Training phases

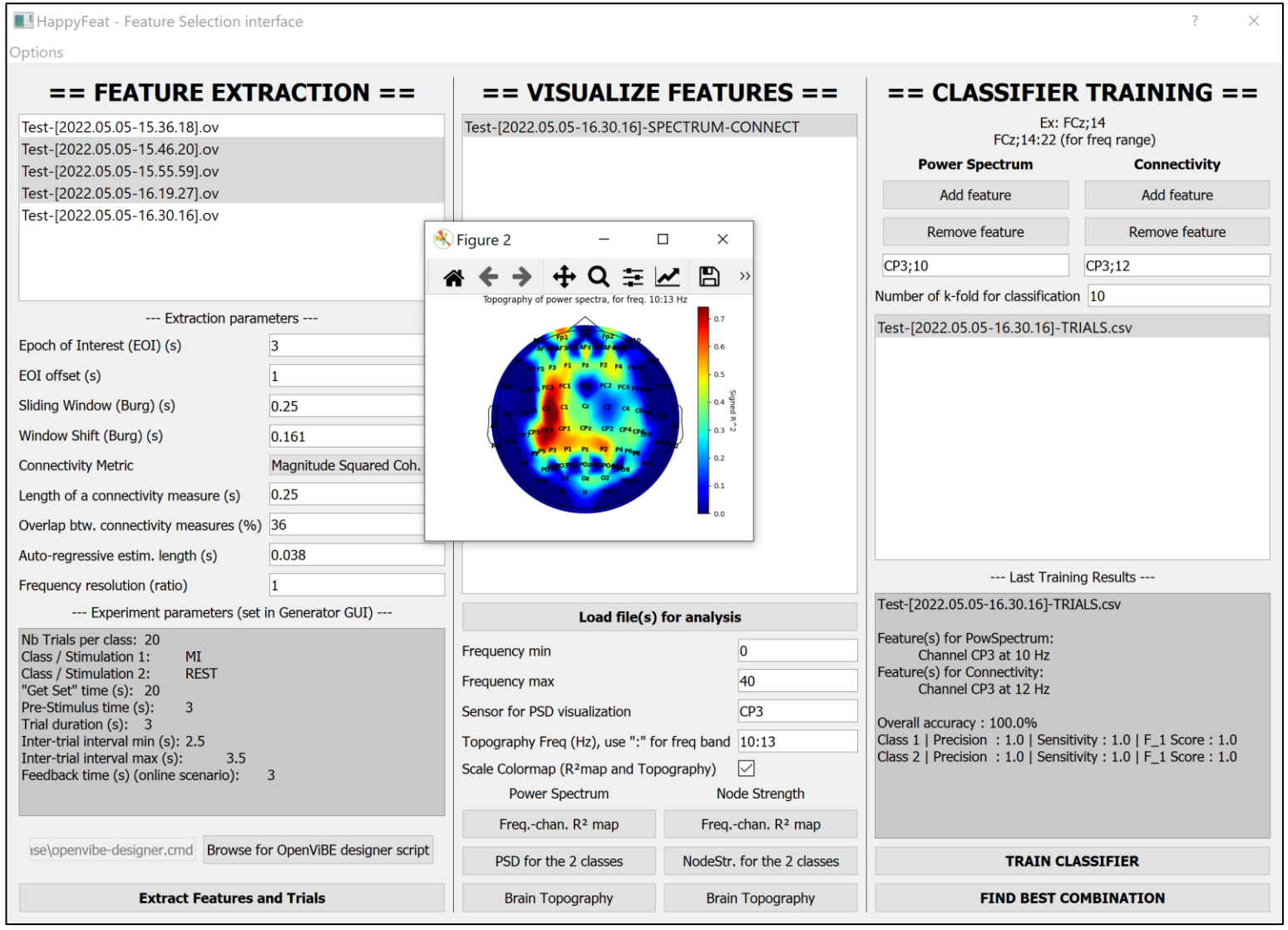
easy & fast

Analyze your data,

select your Features

& train your classifier

in less than 5 minutes!



HappyFeat – Main Concepts

- **Efficient processing pipelines**
  - Available features for classification: *Spectral Power*, *Connectivity-based network metrics*
  - … It's also possible to train the classifier using **a fusion of both features.**
- **Feature extraction**
  - Easy access to all experiment & signal processing parameters.
  - Use **pre-recorded signals** or **on-the-fly during acquisition phase.**
- **Visual Analysis for feature selection**
  - R^2 maps, PSD comparison across trials, time/freq. ERD/ERS analysis, brain topography…
- **Classifier training**
  - **Run as many training attempts as needed**, using different features, in only a few clicks.
  - Concatenate trials from multiple recorded sessions
  - OpenViBE scenarios are **updated and launched on-the-fly**.
  - Automatically generates/updates the **"online classification" scenario**.

HappyFeat – Main Concepts

- **Key feats & mechanisms:**
  - **Clean, risk-free environment**

    ➔ avoid unnecessary & error-prone manipulations.
  - **Trial-and-error oriented workflow**

    ➔ all steps can be repeated quickly & as many times as needed
  - **Unified "dashboard" GUI**
  - **OpenViBE** used in the background, as a fast & efficient processing engine.

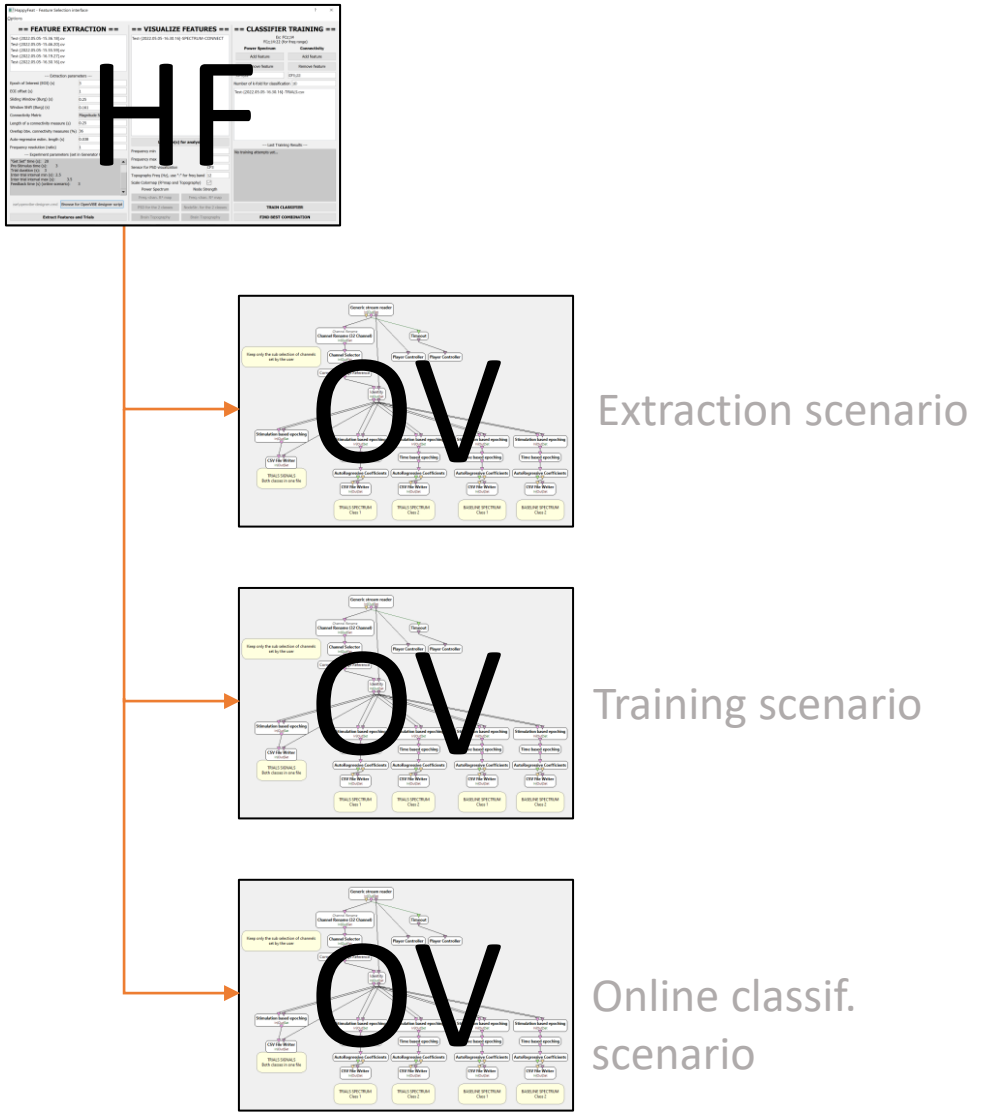
    ➔ no scenario edition/manipulation necessary: everything is **automated**!
- **Two main use cases:**
  - **Make BCI pipelines smoother/easier to use** and allow **reproducibility** of exps.
  - **Prospective works** & **comparison of alternative features of interest** (connectivity, networks…)

HappyFeat - How?

1. **MI Pipeline / “Feature type” selection**

- Selecting btw. multiple “template” scenarios depending on the use case (power spectrum, connectivity type…)

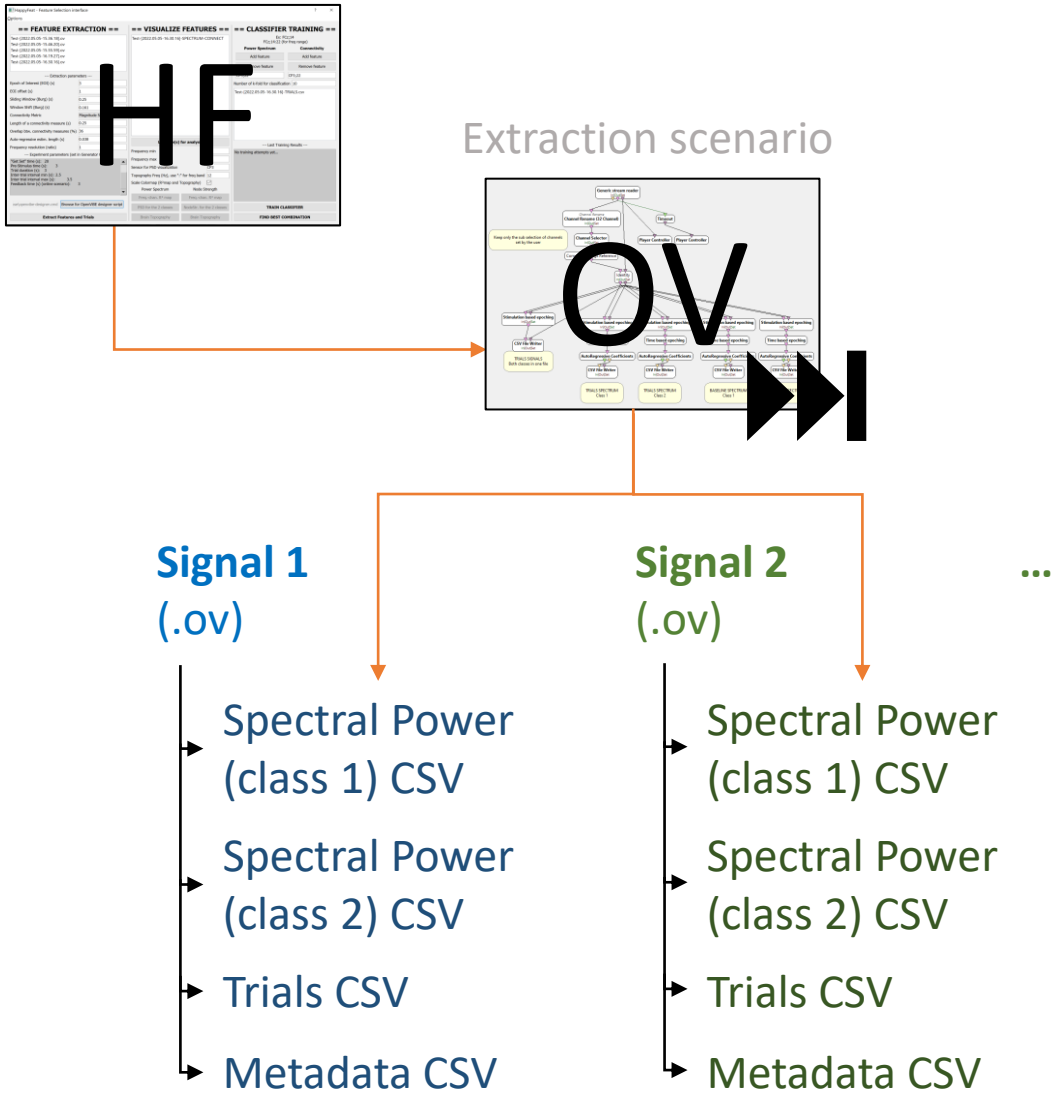
- Edit basic/common parameters (acquisition, extraction, training…)



HappyFeat - How?

1. MI Pipeline / “Feature type” selection
2. **Feature Extraction**

- **Select signal files**, and **extraction parameters** (lengths and overlap of windows, FFT size…)
- **Run** the generated extraction OpenViBE scenario (**in the background**) for all selected signal files:
  - ➔ **Extract metadata** (sampling freq, electrodes…)
  - ➔ Cut the signal to **regions of interest** (MI trials & baseline portions), **generate CSV file** with only these chunks (for the training step)
  - ➔ Apply a **signal processing** pipeline (PSD computation, connectivity measure…) to the signal chunks of interest, **generate CSV files for future analysis**
- **Runs in an autonomous thread:**
  You can do visualizations and training attempts for signals already processed in the meantime.



HappyFeat - How?

1. MI Pipeline / “Feature type” selection
2. Feature Extraction
3. **Analysis, Feature Selection**

- Select one or multiple signals & load their spectral/connectivity data (CSV work files generated during “Feature Extraction”)

- Use different **Visualization Tools** to help find & select **features of interest** (FOIs) for training

  ➔ Frequency/channel R^2 map

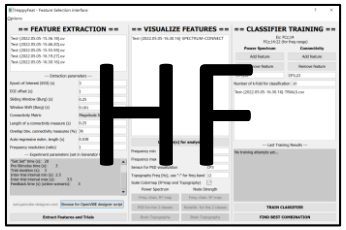
  ➔ PSD (or connect. metric) comparison btw. the 2 conditions (MI/REST) for a given electrode

  ➔ Time/frequency ERD/ERS analysis for each condition

  ➔ R^2 mapped as a brain topography for a given frequency (or range)

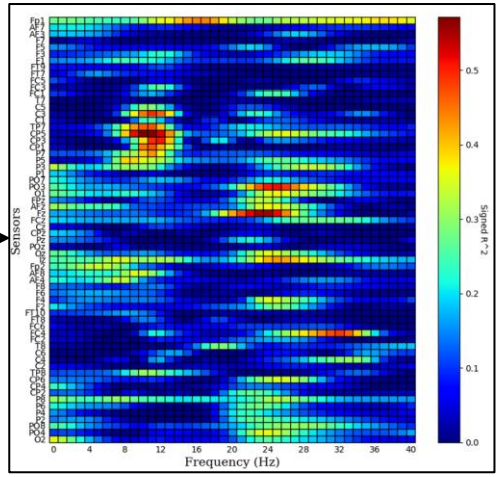
Combine as many visualization windows as you need

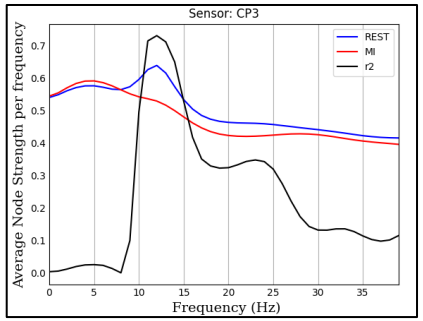
A “Dual metric” pipeline allows to show (for ex.) R^2 maps for both Power Spectrum and Connectivity in parallel

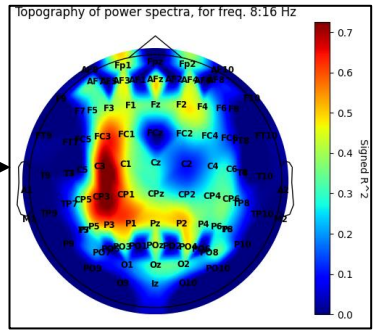


Signal 1
(.ov)

- Spectral Power (class 1) CSV
- Spectral Power (class 2) CSV
- Trials CSV
- Metadata CSV







HappyFeat - How?

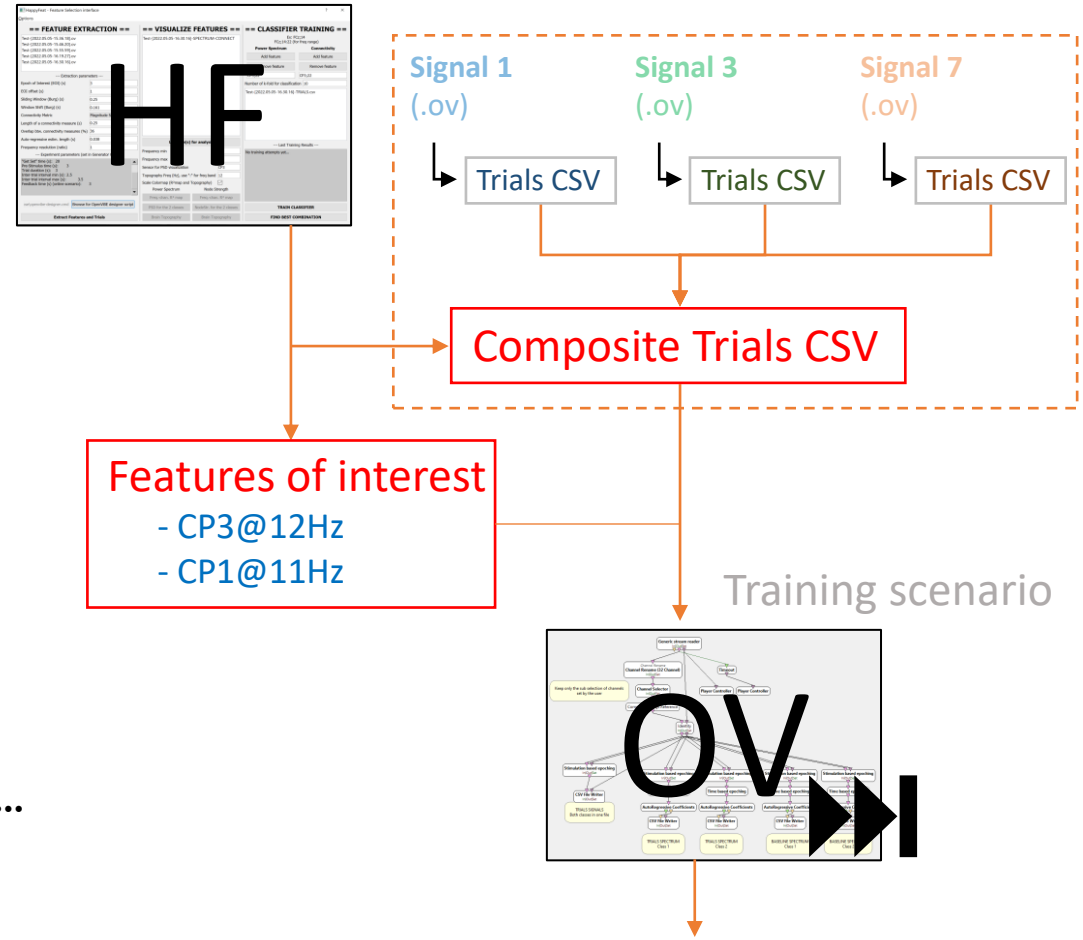
1. MI Pipeline / “Feature type” selection
2. Feature Extraction
3. Analysis, Feature Selection
4. **Classifier Training**

- Set one or more **Features of Interest** (Electrode/Freq.)
- **Select file(s)** with which you want to train your classifier
  If > 1 file : their trials **are automatically concatenated**
- **Run the Classifier Training scenario**
  (Auto. generated in step 1 + auto. edited with FOIs)
  ➔ Classification **ACCURACY** + **WEIGHTS**

Disappointing results? (“My accuracy is 50% ☹”)
Maybe try again with other features. **It only takes a few seconds…**

Satisfying results?! (“OMG 95%”)
Good news! The “Online Classification” scenario has already been automatically been updated with:

- Classifier training weights
- Features of interest used for training



Training accuracy “**score**”
+ Classifier **Weights**
+ **Online scenario updated**

A few technical details…

Python framework

- GUI (PyQt5), automations (xml scenarios, json, numpy…), data processing (MNEPython,…), visualizations (matplotlib, soon seaborn & visbrain), threading

Full list of dependencies:

- **Python 3.9**
  - shutils>=0.1.0
  - mne>=0.23.0
  - numpy>=1.21.1
  - pandas>=1.3.1
  - PyQt5>=5.15.6
  - statsmodels>=0.13.1
  - scipy>=1.7.1
  - spectrum>=0.8.0
  - matplotlib>=3.4.2

- **OpenViBE v3.4.0** (coming Dec.2022) ➔ for now, limited to Windows & Unix

Yet more technical details…

- **Type of processed data**
  - EEG signals (also compatible with MEG in the foreseeable future)
    from OpenViBE (.ov format)

- **Computational resources:**
  - Fully autonomous on a single computer
  - No server/client solution
    (except if running OpenViBE in server/client for acquisition!)

Software ecosystem

- **Development team: 1 (☺)**
  - Lots of help (on data analysis & visualization, debugging, setting up scenarios…) from Tristan Venot, who uses the software daily for experiments in the context of his PhD. Many thanks to him!

- **Expected users:**
  - BCI users (researchers, PhD students, clinicians…)

    … *with or without technical/programming skills*
  - Possible contributions: mainly in setting up pipelines & suggesting new metrics

Current limitations & future works

- **Current limitations**
  - Low flexibility regarding electrode schemes. *(work in progress…)*
  - Only one type of classification algo. proposed (LDA) *(work in progress…)*

  - Pipelines are “fixed”:
    — trading OpenViBE’s high level of flexibility…
    + …for a high comfort of pipeline settings customization
    + & “trial-and-error” workflow
  - Only three types of pipelines/feature types:
    - Power Spectrum Density
    - Connectivity (coherence & its variants)
    - Dual (mixing PSD & Connectivity)

  ➔ In project (*long term!*) for more “**prospective power**”:
  allow the user to choose btw. 1 and 3 feature-types & network metrics
  to mix as they see fit (MSC/node strength + Imag(Coh)/Laterality + …)

Current limitations & future works

- **Current limitations**
  - Every time a new feature type is selected, or extraction parameters are modified…

    all work files need recomputating from scratch.

    ➔ Necessity for a robust “work session” save/load mechanism

  - In project: fully autonomous 100% Python version, without OpenViBE

    — No acquisition/online possibilities

    — Obviously slower… (no C++ optimizations!)

    \+ More portable, all types of platforms supported (MacOS!)

    \+ Other (more flexible) formats for I/O and work files (CSV, EDF…)

HappyFeat – Conclusion

- **Already available online, work-in-progress version:**
  https://github.com/Inria-NERV/happyFeat
- **To be continued…**
  - **More flexibility** (pipeline options, mixing metrics…)
  - **More network metrics** (based on connectivity)
    - **… and associated visualization tools**
  - **More options for classification algorithms**
  - **Workspace/session manager** to save/load session settings
  - Fully autonomous Python version, for cross-platform usage

First official release early 2023 – stay tuned!
(open-source license to be determined…!)

BCI Motor Imagery with OpenViBE in X-Men: First Class

Thanks for your attention! Any questions?