

NEC24 Workshop 1 - July 8th 2024

**HappyFeat: an interactive and
efficient BCI framework
for clinical applications**

Arthur Desbois

Inria Paris, NERV team, Paris Brain Institute (ICM)

1. Context

- What is BCI? What are we looking for?
- Feature-based classification vs. BCI variabilities

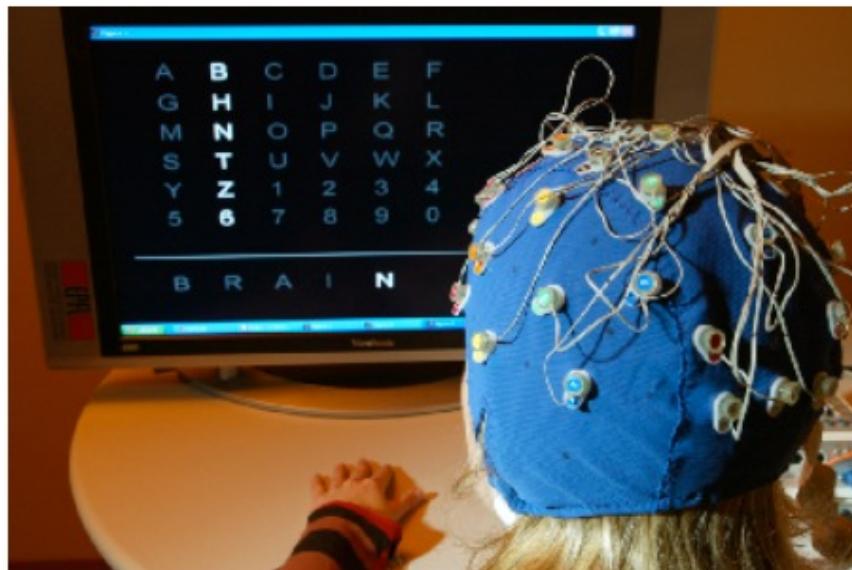
2. HappyFeat

- Key points
- Example through typical use-case

3. Perspectives & conclusion

1. Context

- What is BCI? What are we looking for?
- Feature-based classification vs. BCI variabilities



<http://mmspgo.epfl.ch/research>

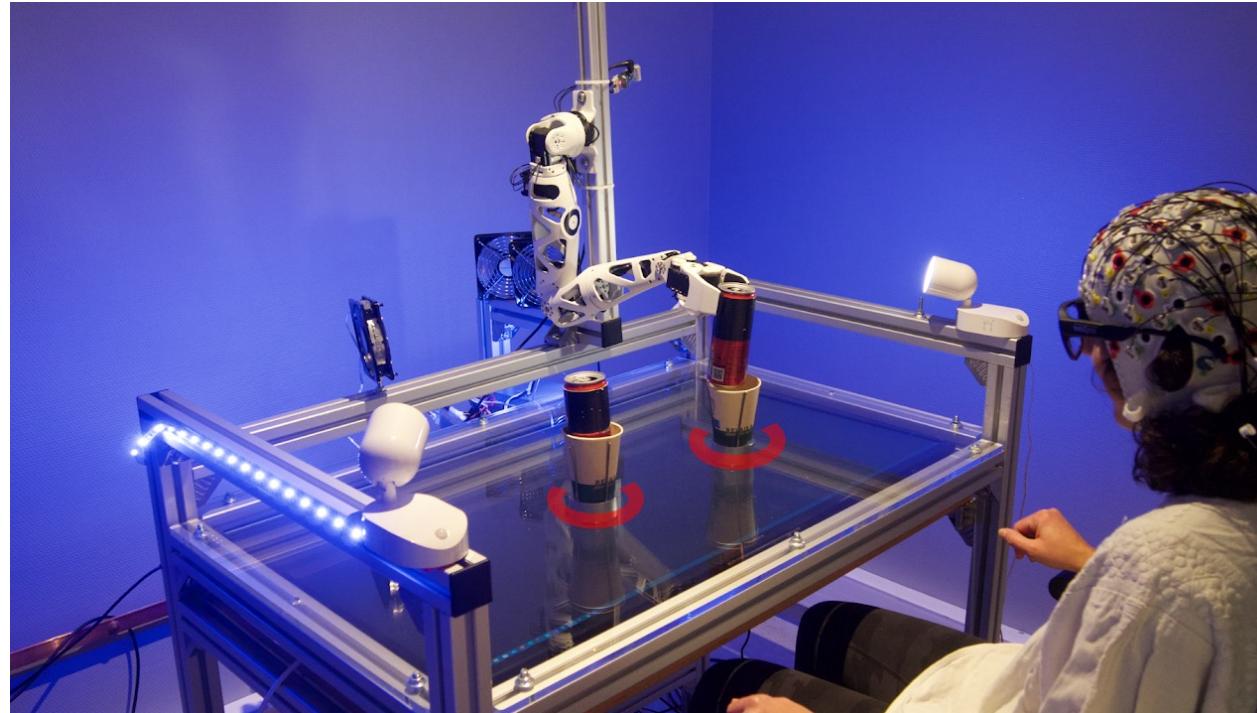


SSVEP – Inria/Hybrid

What is “BCI”?

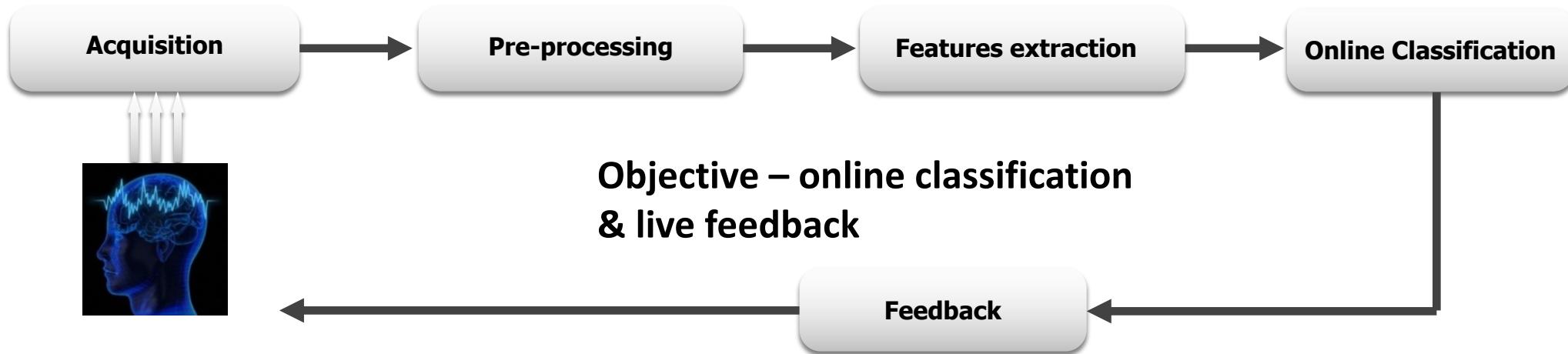


Inria

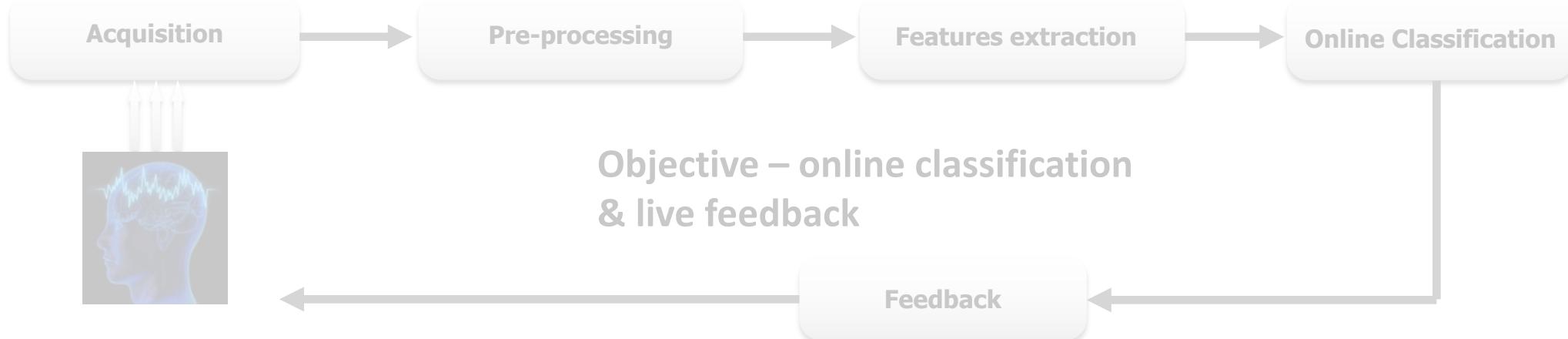
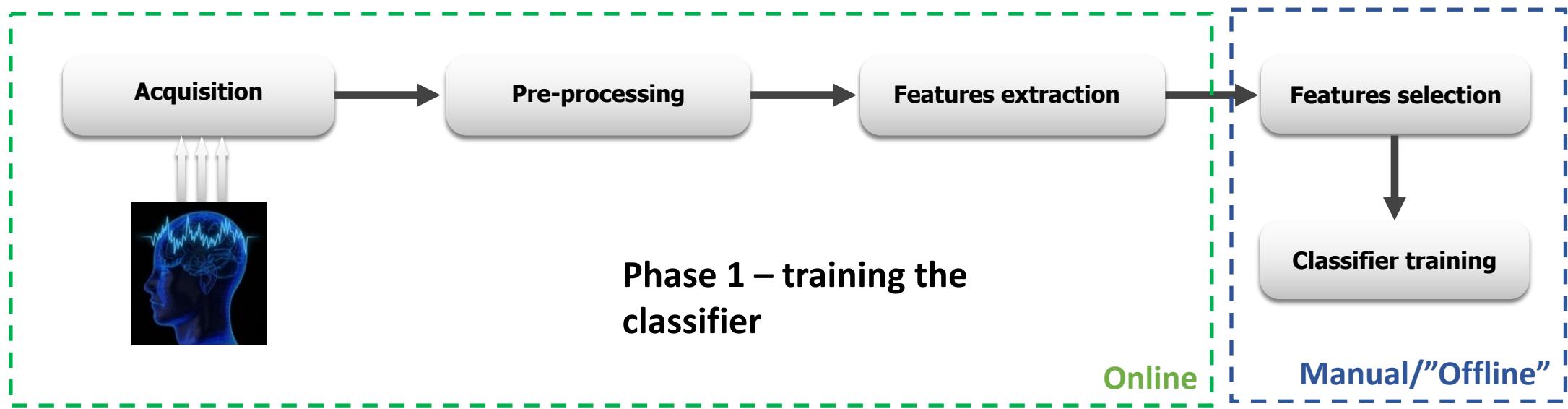


Protocole BRACCIO – Venot et al., ICM/NERV

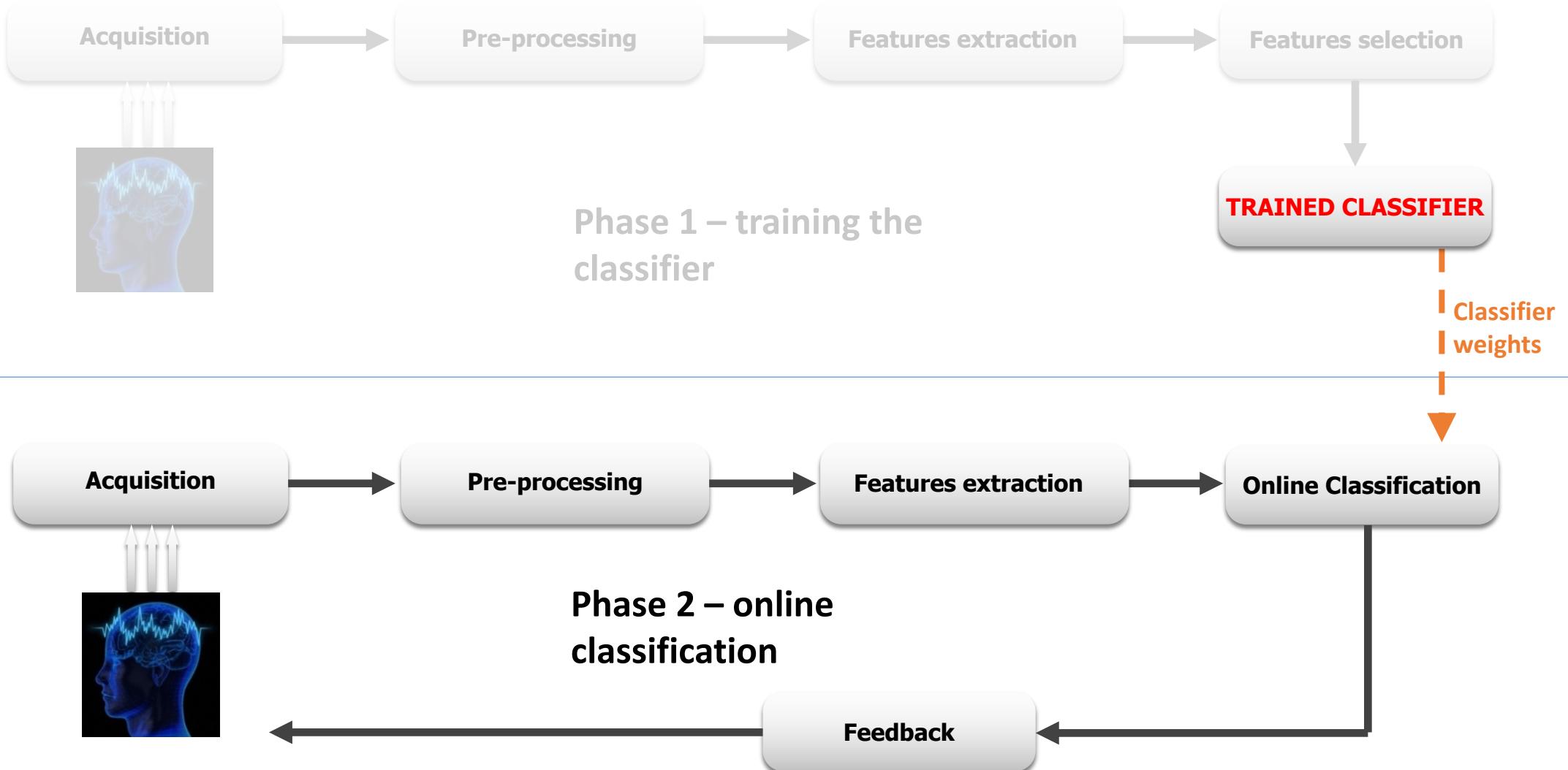
What is “BCI”?



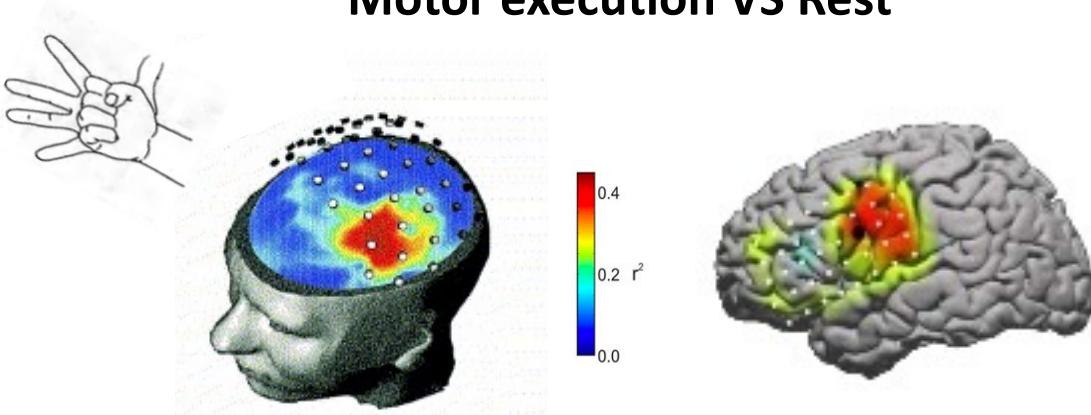
What is “BCI”?



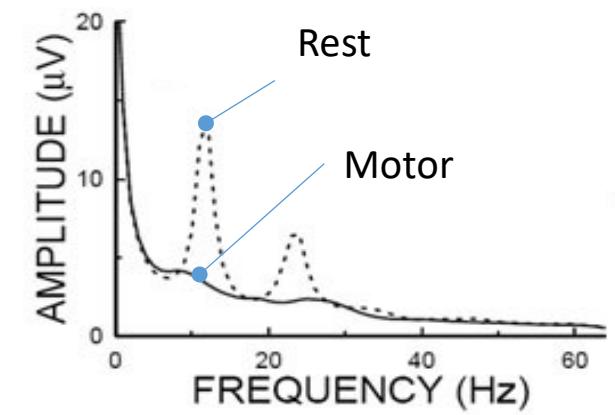
What is “BCI”?



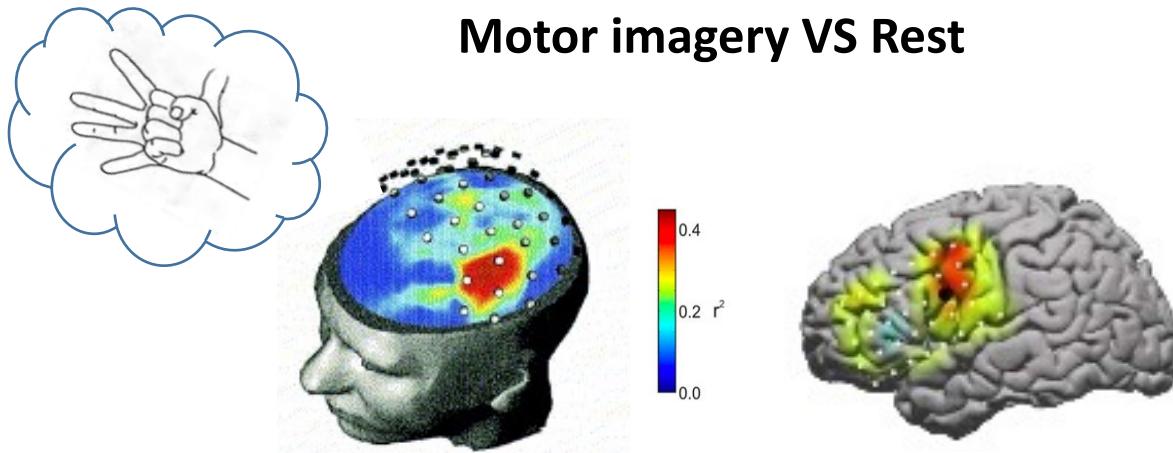
Motor execution VS Rest



Spectral Power Decrease



Motor imagery VS Rest

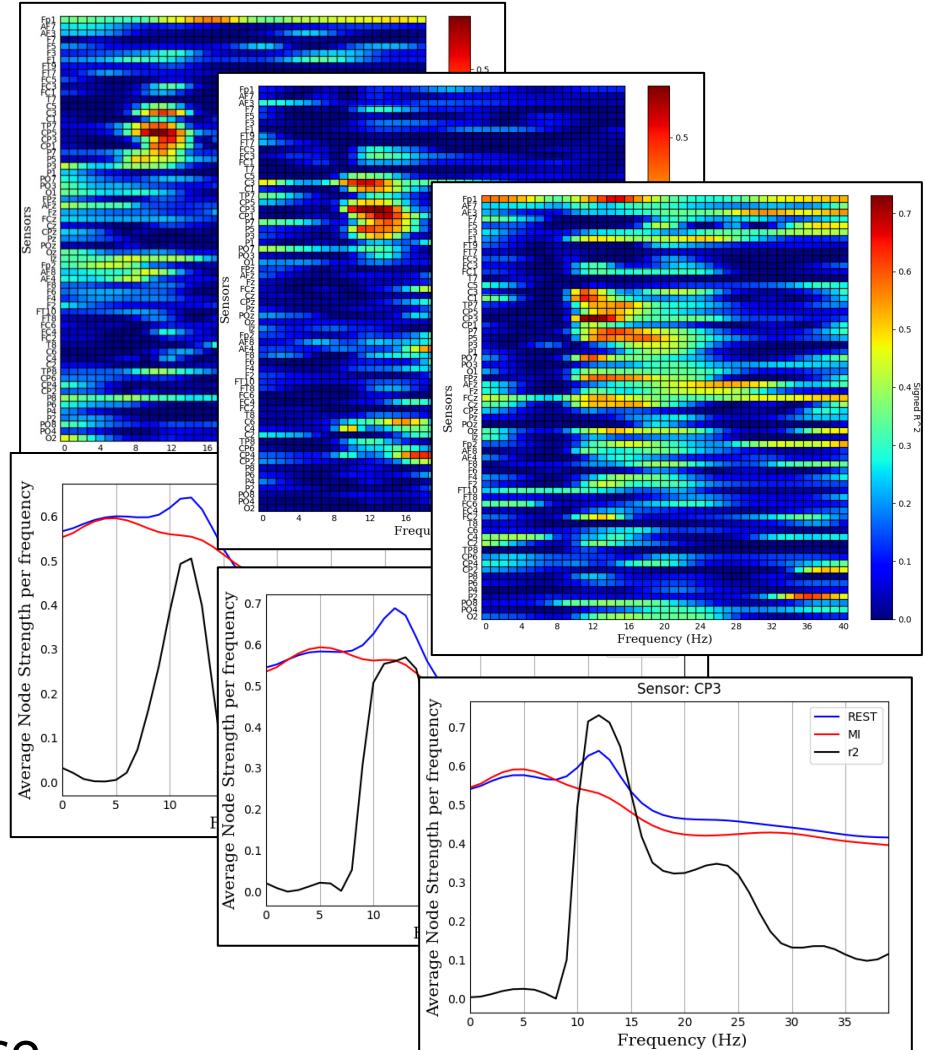


Desynchronization effect
(Pfurtscheller et al, 1999)

- **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...

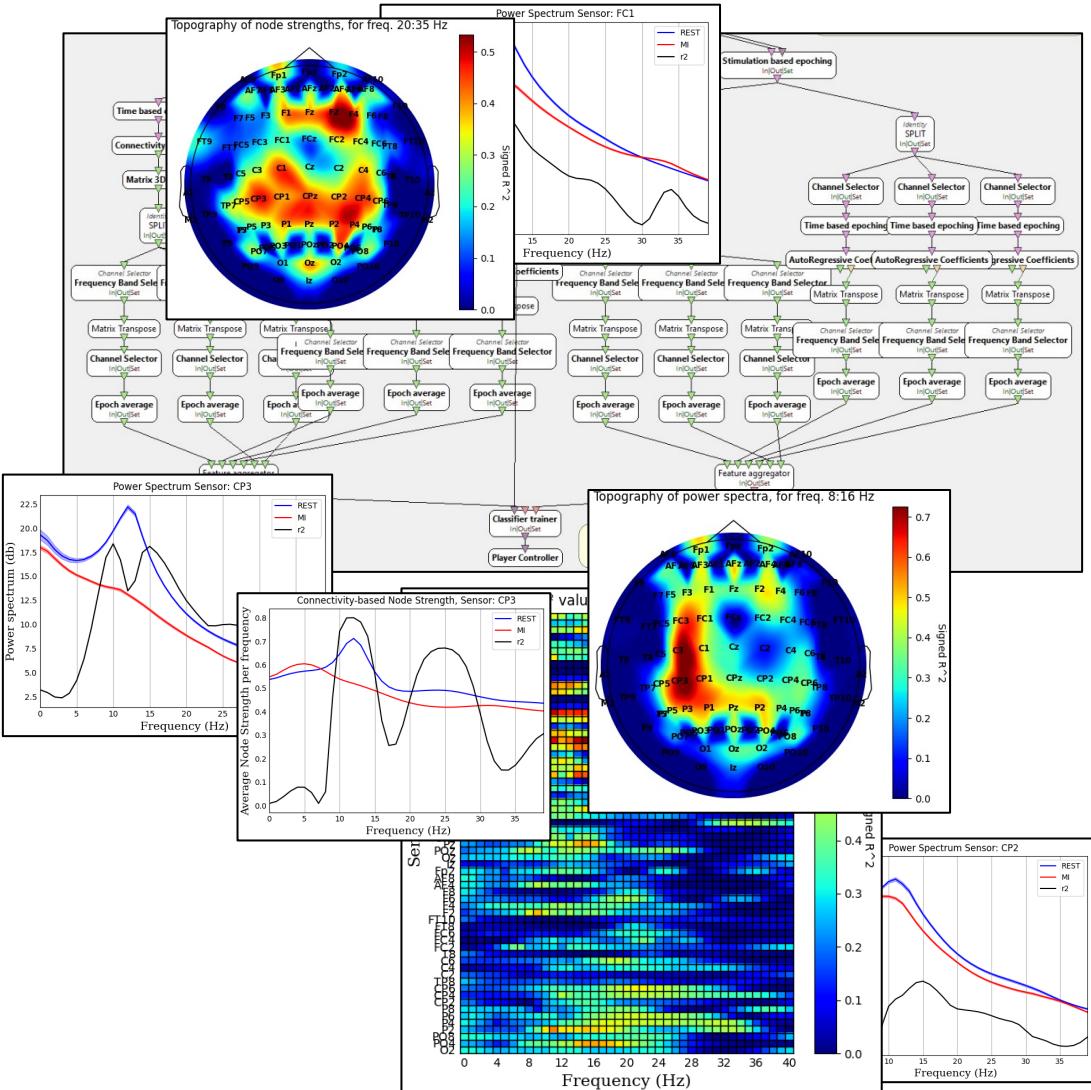


BCI in clinical settings - Feature selection



The “analysis phase” involves many manipulations. An example:

- Setting up “feature extraction” pipelines (e.g. OpenViBE)
- Finding FOIs through visualization (e.g. with Matlab, Python, etc.)
- Setting up & running training pipelines (e.g. OpenViBE)
- ... and maybe going again through those steps multiple times until “correct” features have been found, or to account for inter-run variability



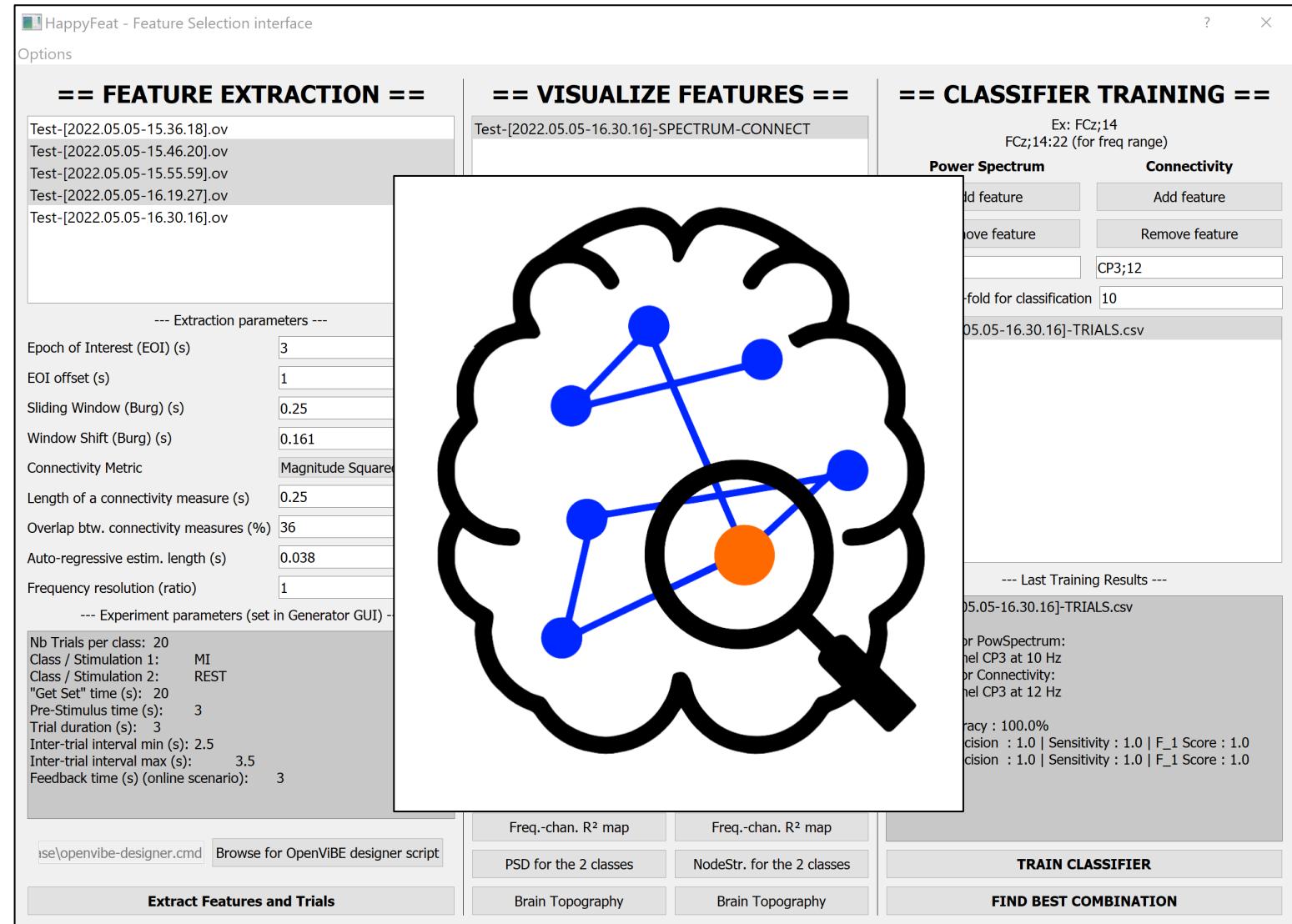
→ Tedious, error-prone,
hard to achieve in a limited time

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!



1. Context

- What is BCI? What are we looking for?
- Feature-based classification vs. BCI variabilities

2. HappyFeat

- Key points
- Example through typical use-case

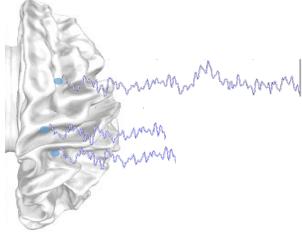


Unified, dashboard-like GUI

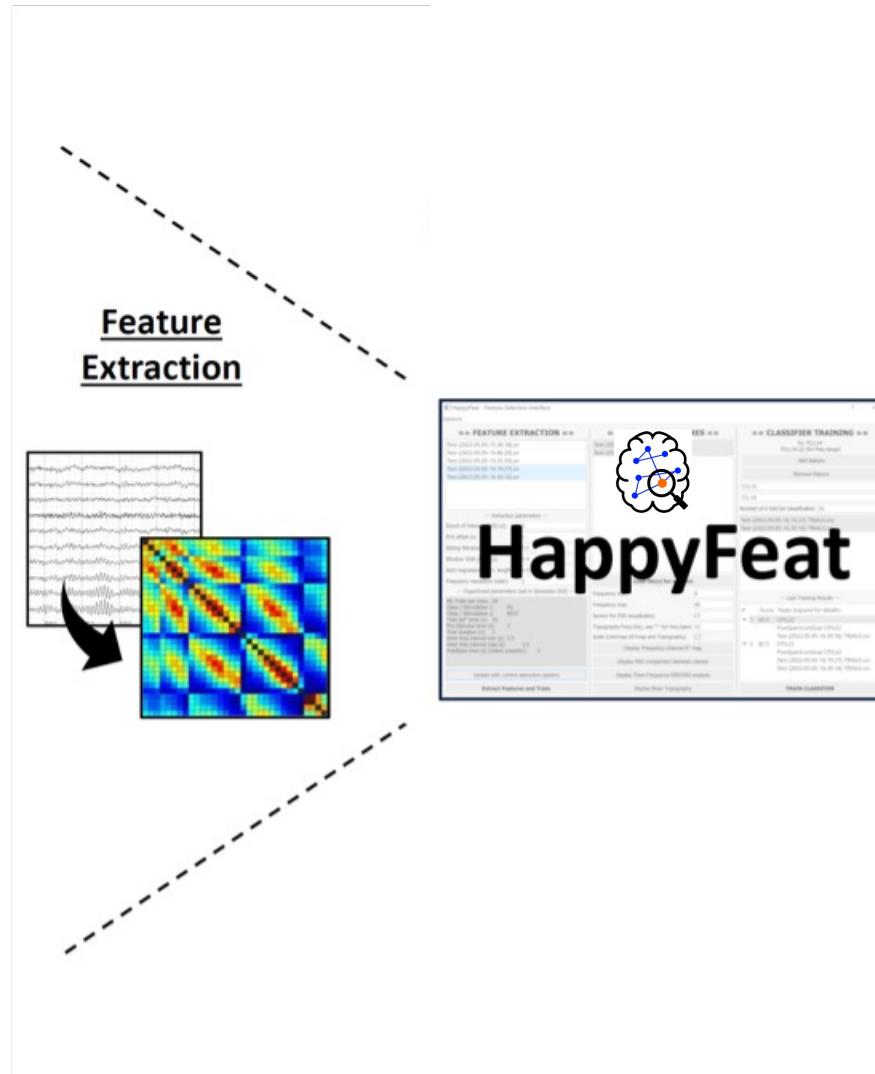
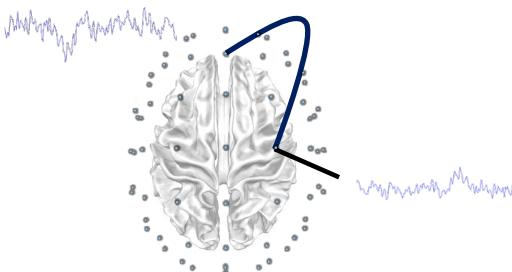
Key-Features

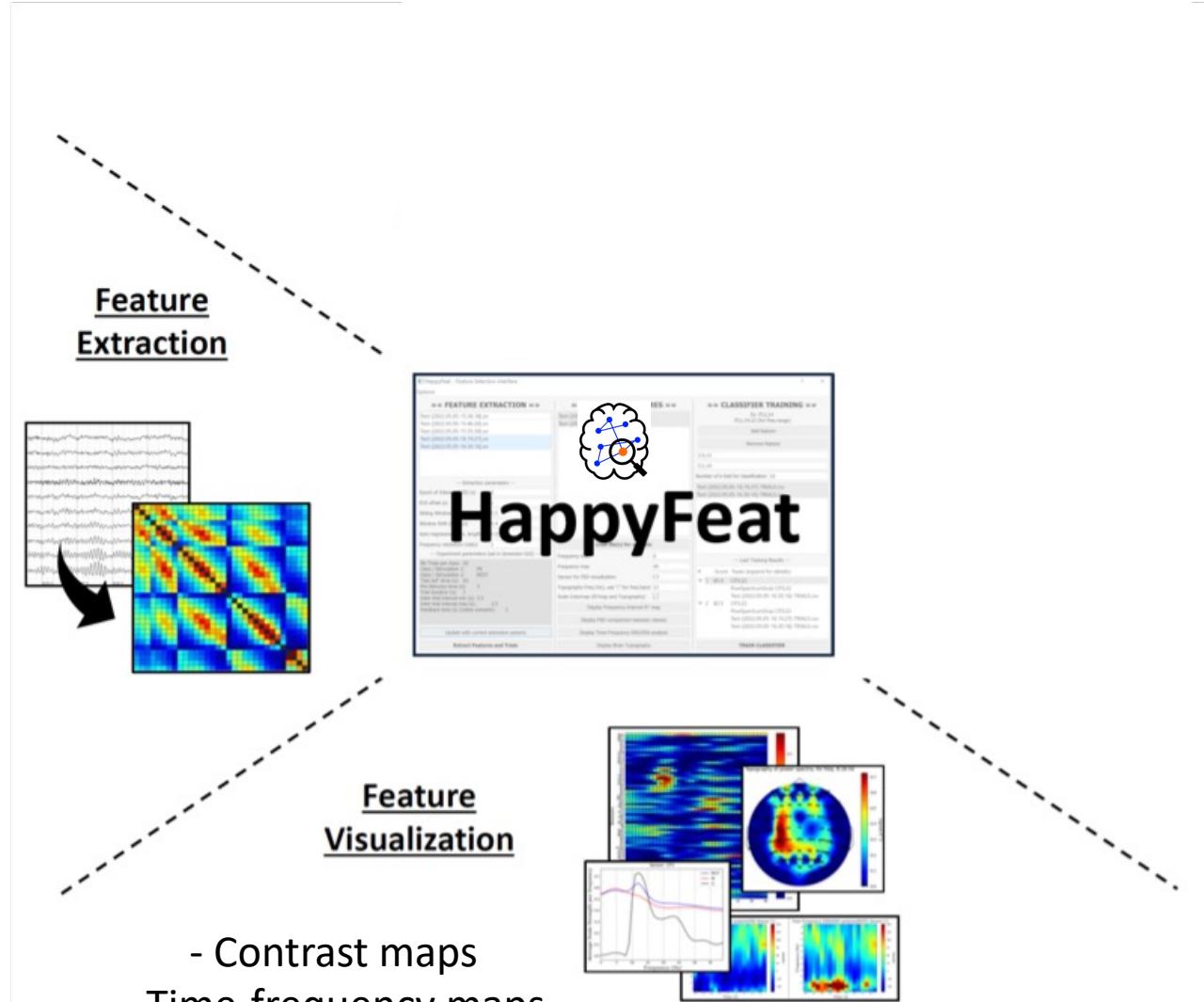


- Classical local measures

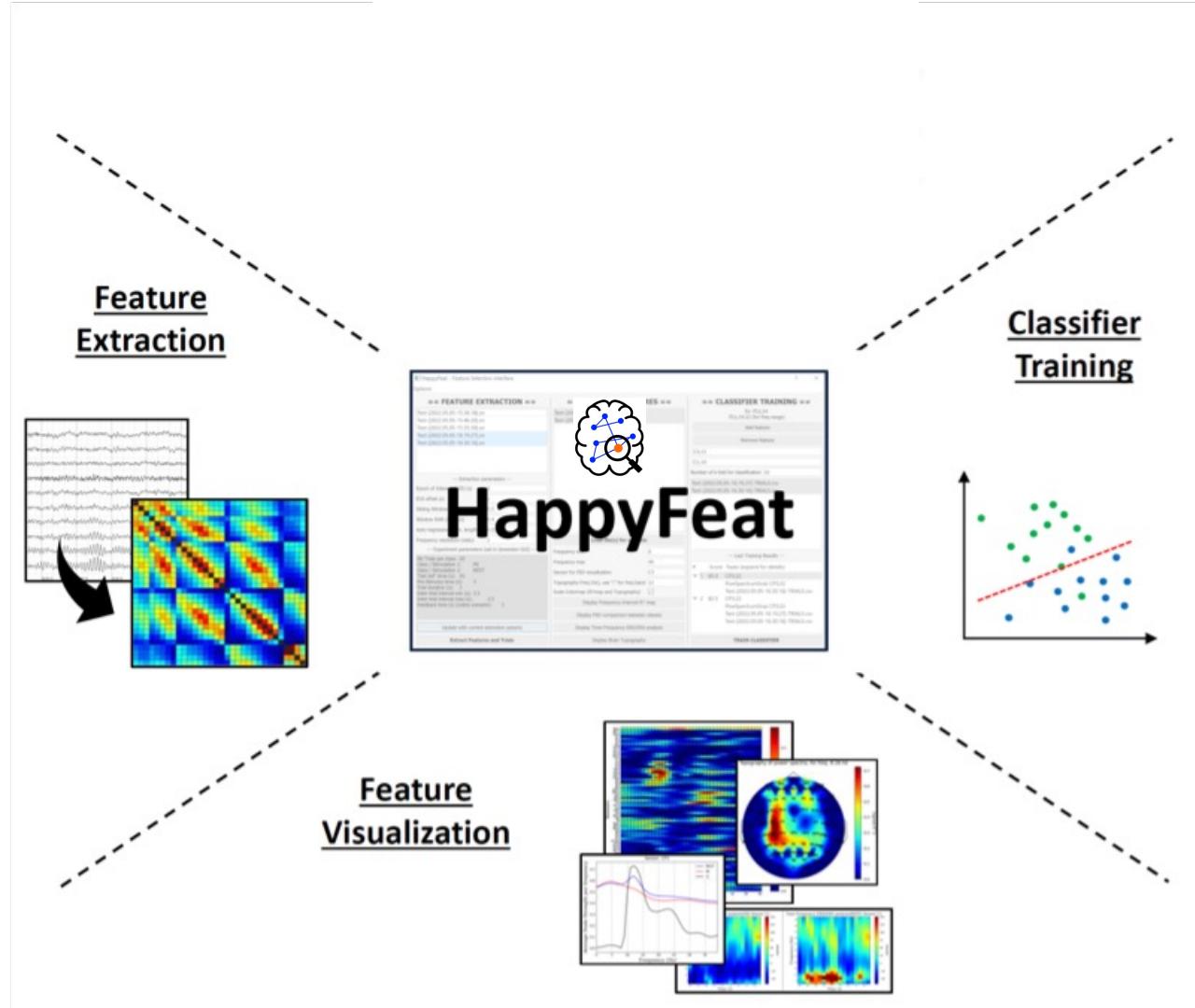


- Novel network-based
approaches





- Contrast maps
- Time-frequency maps
- Topographies



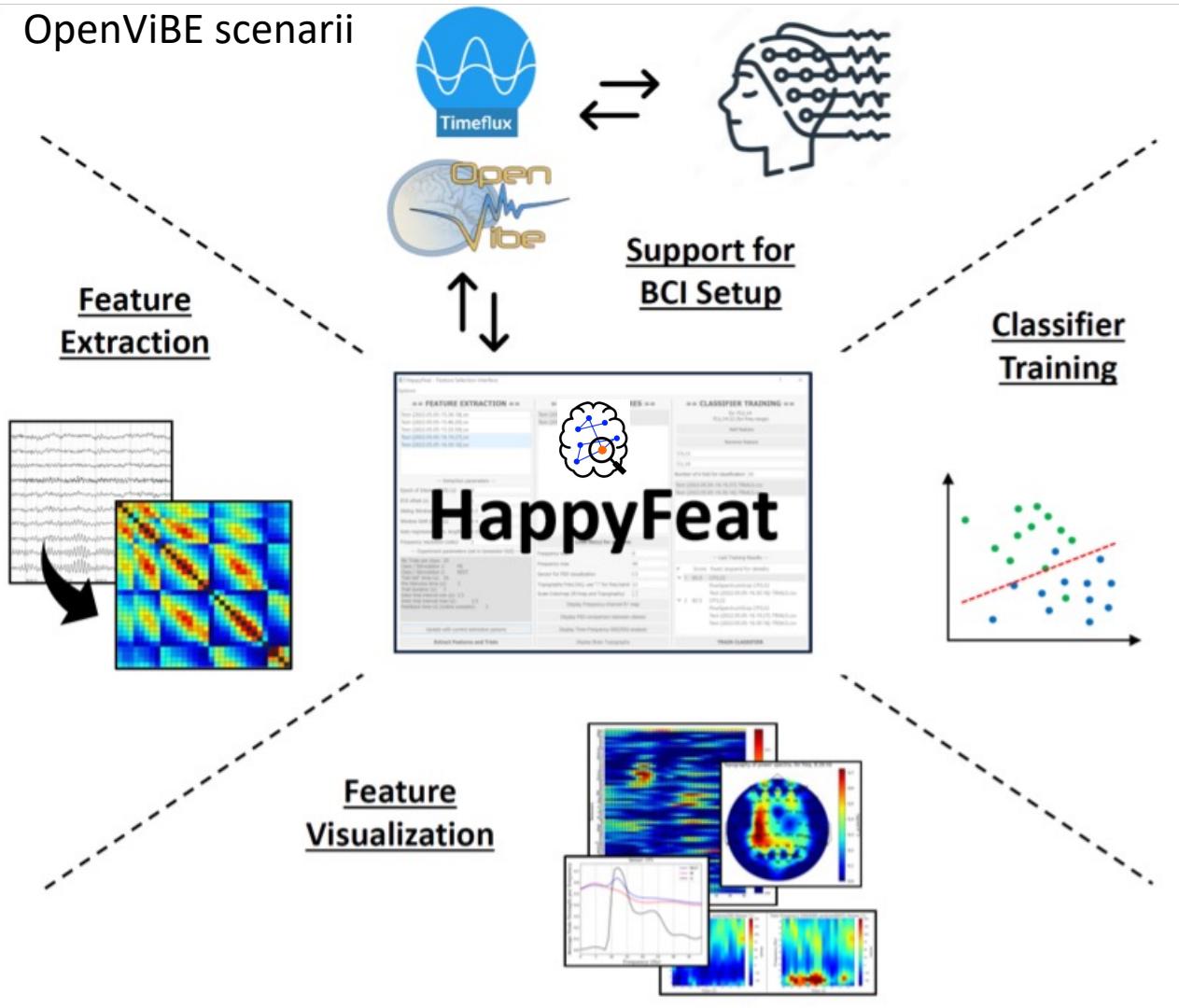
Tool for the decision

Key-Features

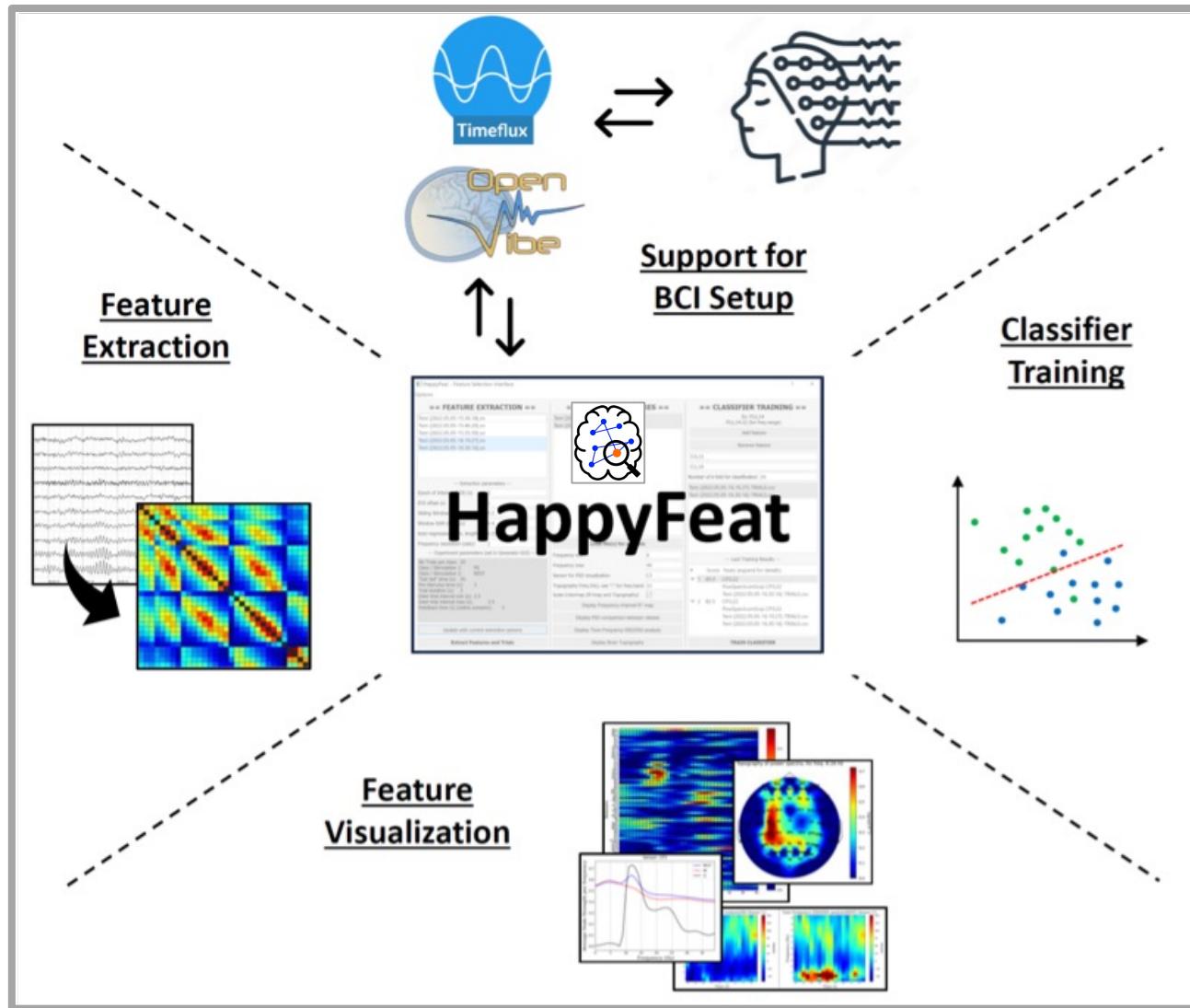


Automatic manipulation of

OpenViBE scenarii



Key-Features



**Workspace to ensure
reproducibility and
replicability of all the
manipulations**

- **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 + Workspace management system**

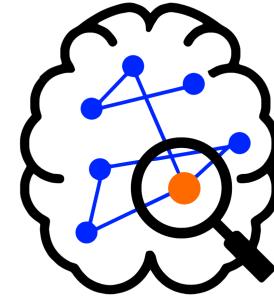
- **Direct link with BCI software(s): OpenViBE (TimeFlux in the near future!)**

→ no scenario edition/manipulation necessary: everything is **automated!**

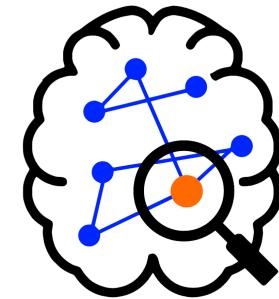
- **Two main use cases:**

- **Make BCI pipelines smoother/easier to use and allow reproducibility of exps.**

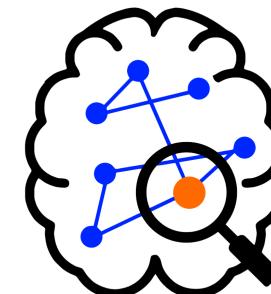
- **Prospective works & comparison of alternative features of interest (connectivity, networks...)**



- **Efficient processing pipelines**
 - Available features for classif.: *Spectral Power (PSD)*, *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.
- **Assisting feature selection: Visualization, AutoFeat**
 - Viz: R^2 maps, PSD comparison across trials, time/freq. ERD/ERS analysis, brain topography
 - Select features manually using visualization tools...
....or a set of the channel/freq couples with the highest R^2 , **automatically**



- **Classifier training**
 - Run as many training attempts as needed, using different features, in only a few clicks.
 - Concatenate trials from multiple recorded sessions
 - Automatically try different feature combinations with one click
- **Onwards to online classification**
 - Automatically generate/update the “online classification” OpenViBE / TimeFlux scenario using selected classification feats. and trained classifier.

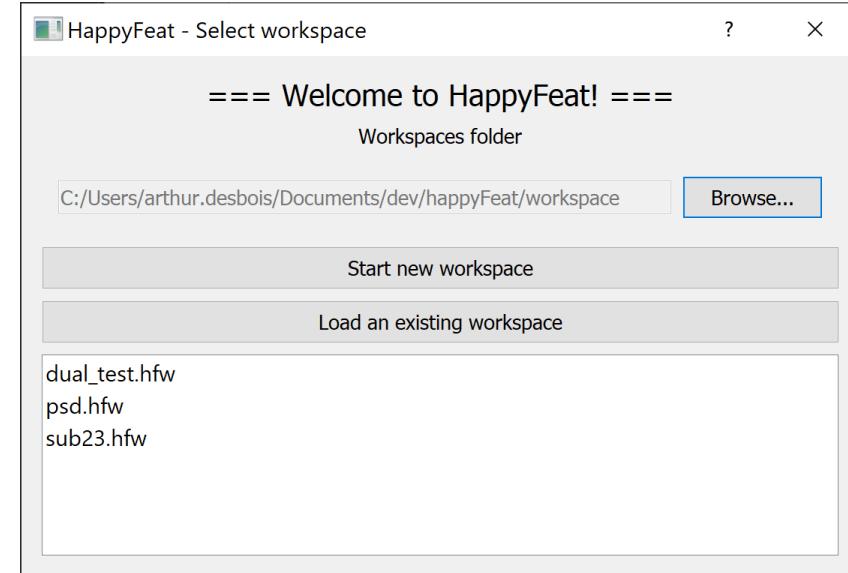


1. Workspace creation/selection

- Create a new environment (file tree, configuration, data structures) for the experiment/analysis

Ex: **sub23_session01**

- ... or select an already existing one.



➔ All user settings & manipulations are saved and available in the workspace for future usage or sharing:

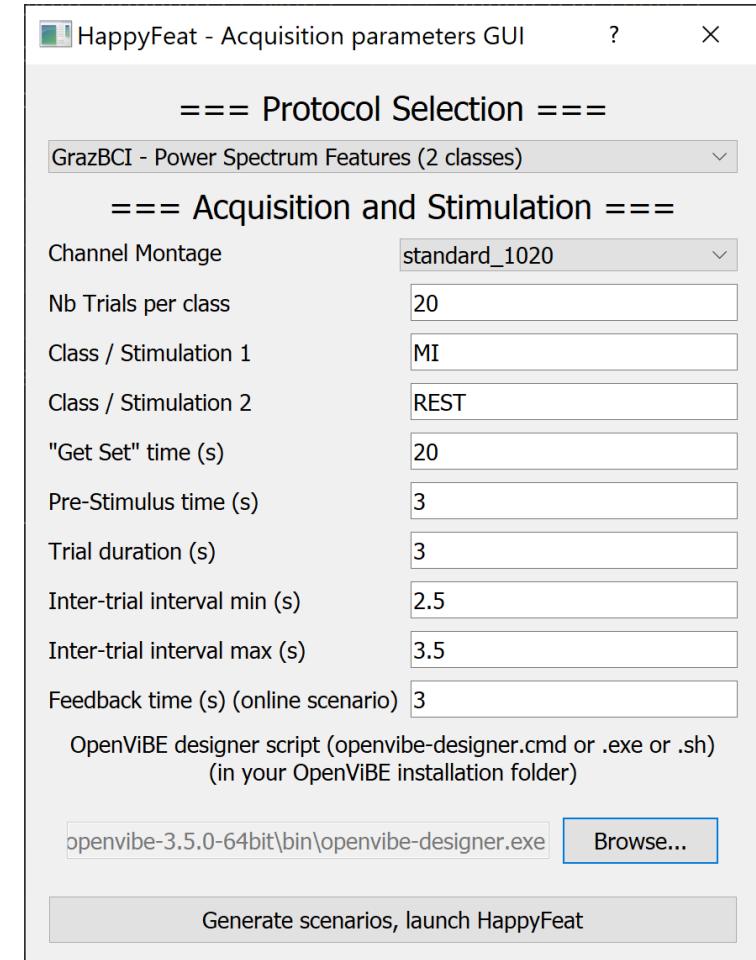
- Extraction parameters (spectral analysis, list of processed files...)
- Training attempts (runs & features used, training accuracy...)

1. Workspace creation/selection

- When creating a new workspace:
 - the feature type (PSD, Connectivity...) is chosen
 - experimental/acquisition parameters are entered (electrode montage, stimulation names, experimental layout...)

➔ This step allows to automatically set up the OpenViBE acquisition scenario.

These experimental parameters are also important for the different processing steps of HappyFeat.



HappyFeat - Typical usecase



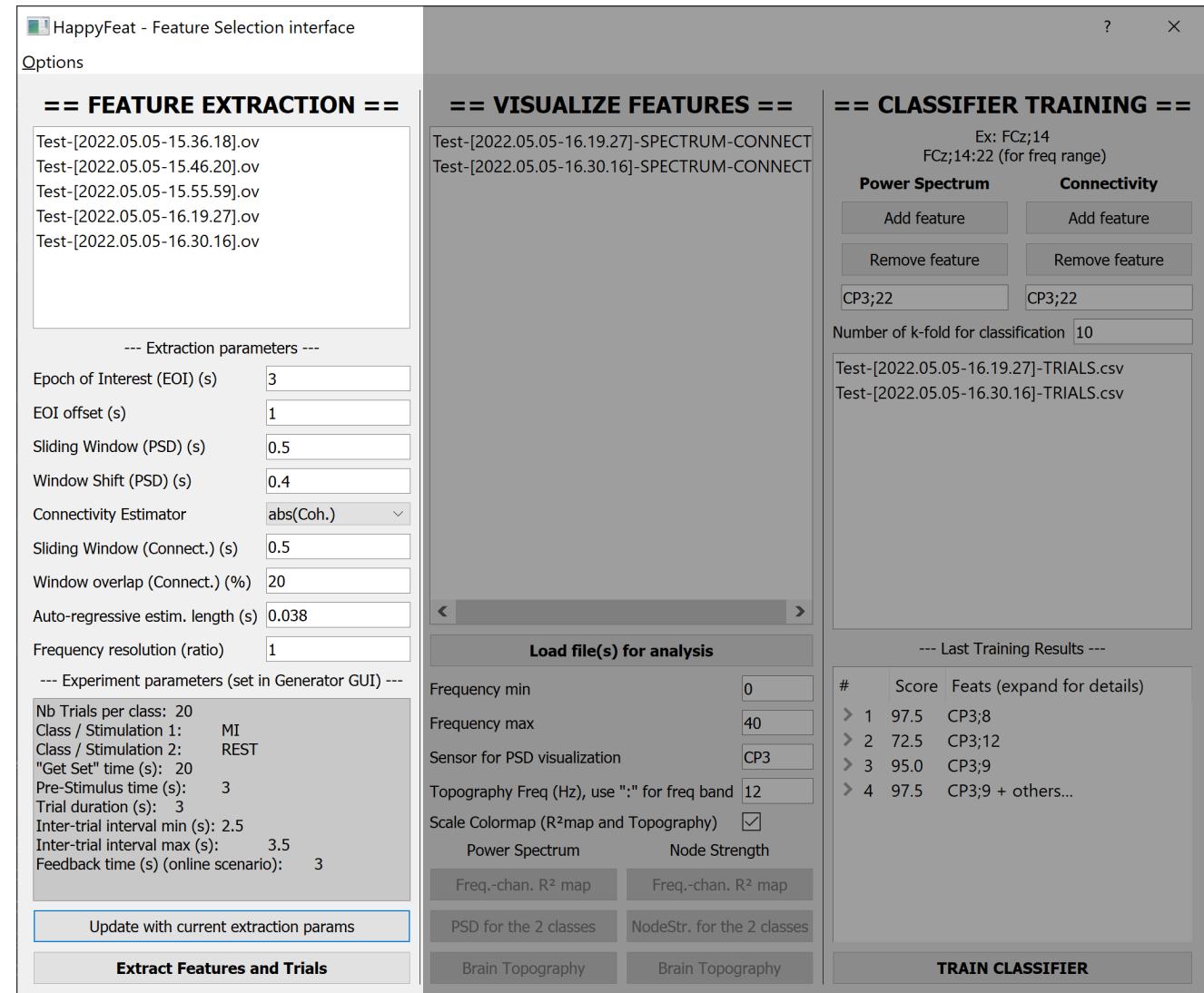
1. Workspace creation/selection

2. Feature Extraction

- Files in the “signals” folder of the workspace are listed in the left panel
- Select one or multiple files...
- Enter the parameters for feature extraction...
- Press the button!

 Along the **workspace** mechanism, HappyFeat also uses **work sessions** to keep track of extraction parameters.

Therefore, all steps realized with a set of extraction parameters (which files have undergone extraction + training attempts) are stored and can be recovered.

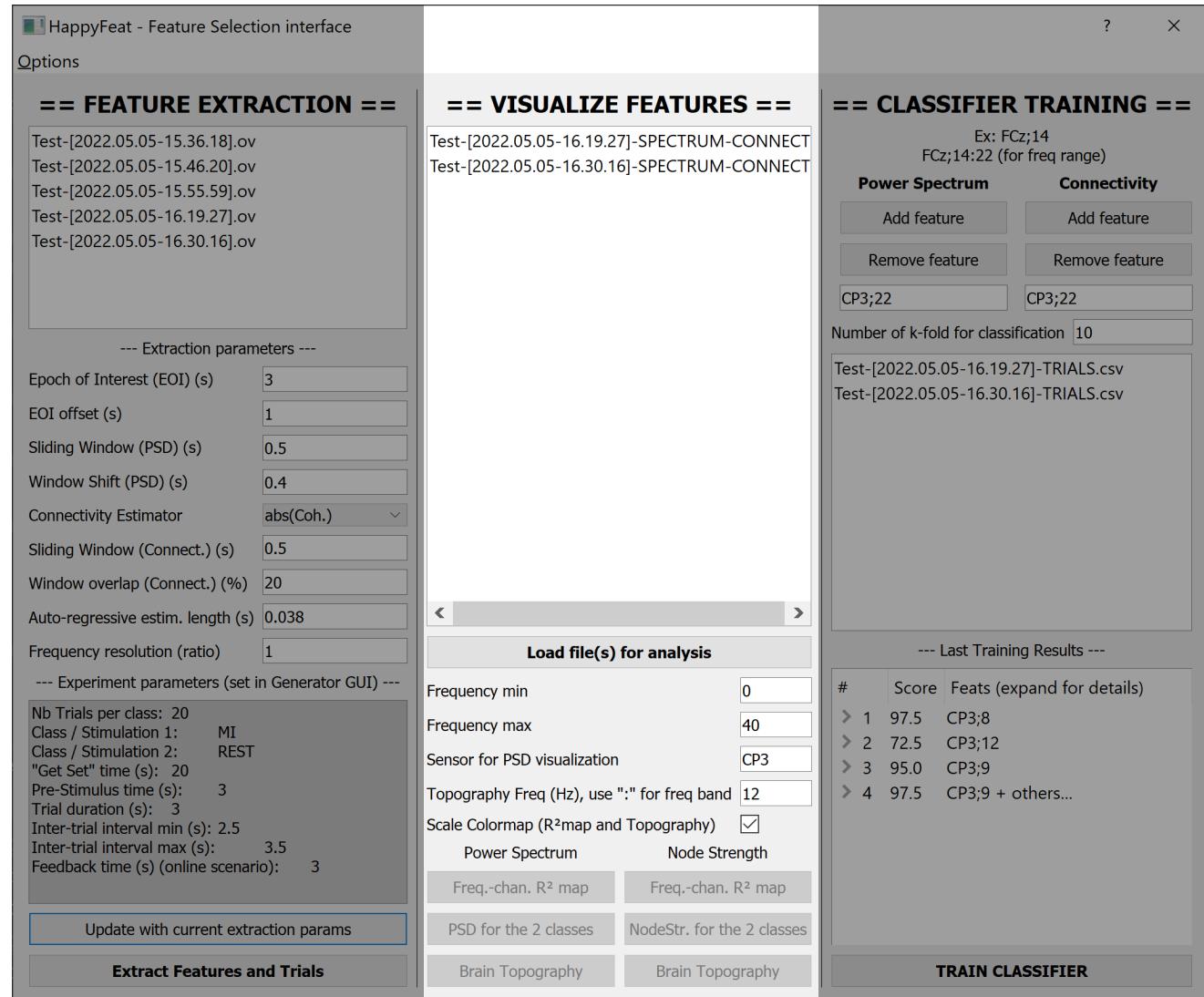


HappyFeat - Typical usecase



1. Workspace creation/selection
2. Feature Extraction
3. Analysis & Feature Selection

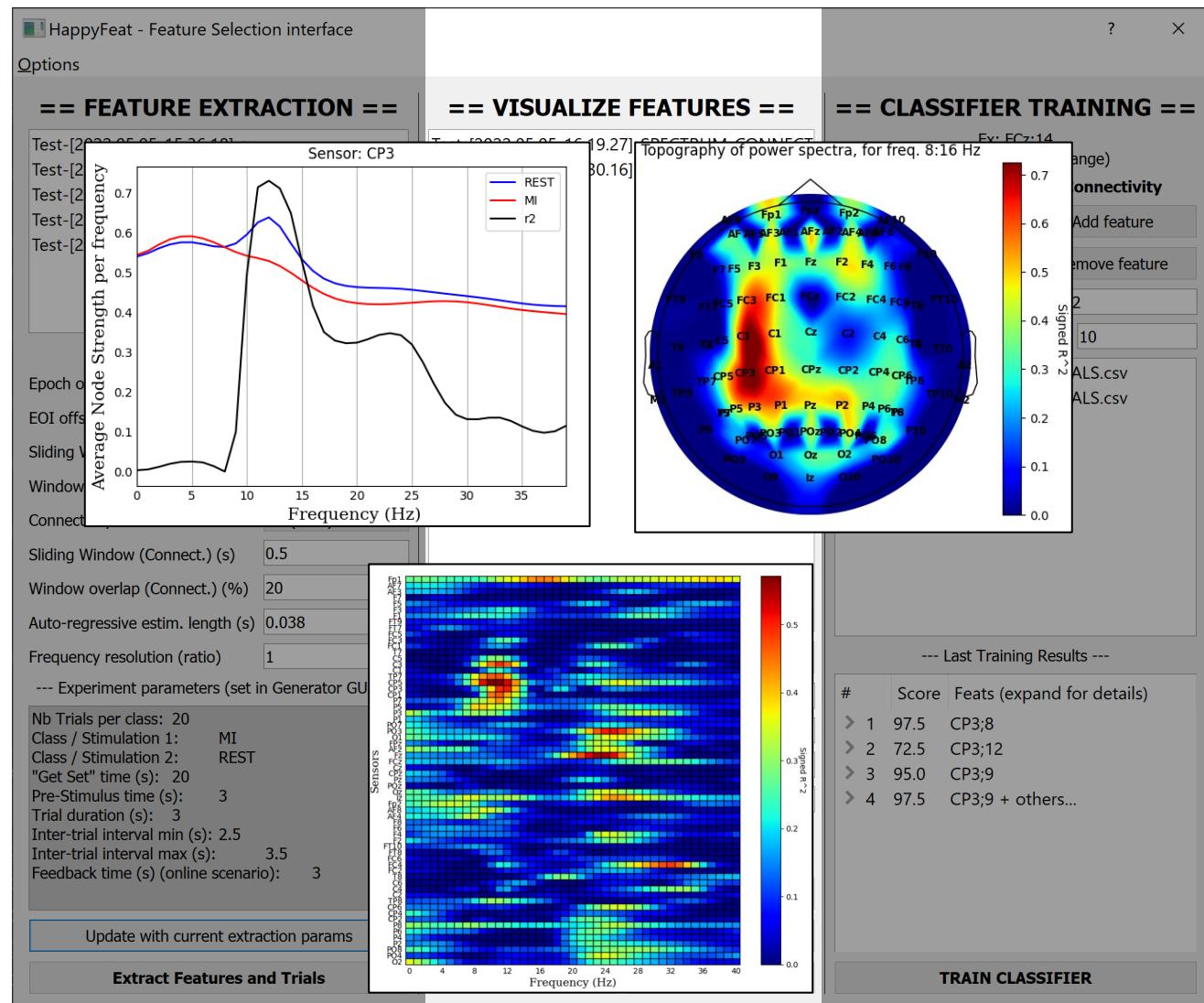
- “Extracted” files (runs) appear in the central panel.
- Select which runs to analyze...



HappyFeat - Typical usecase

1. Workspace creation/selection
 2. Feature Extraction
 - 3. Analysis & Feature Selection**

- “Extracted” files (runs) appear in the central panel.
 - Select which runs to analyze...
 - And use the different **visualization tools** to select adequate features:
 - Channel/frequency R^2 map
 - Per-channel metric comparison
 - Per-frequency R^2 topography
 - More coming soon...



HappyFeat - Typical usecase

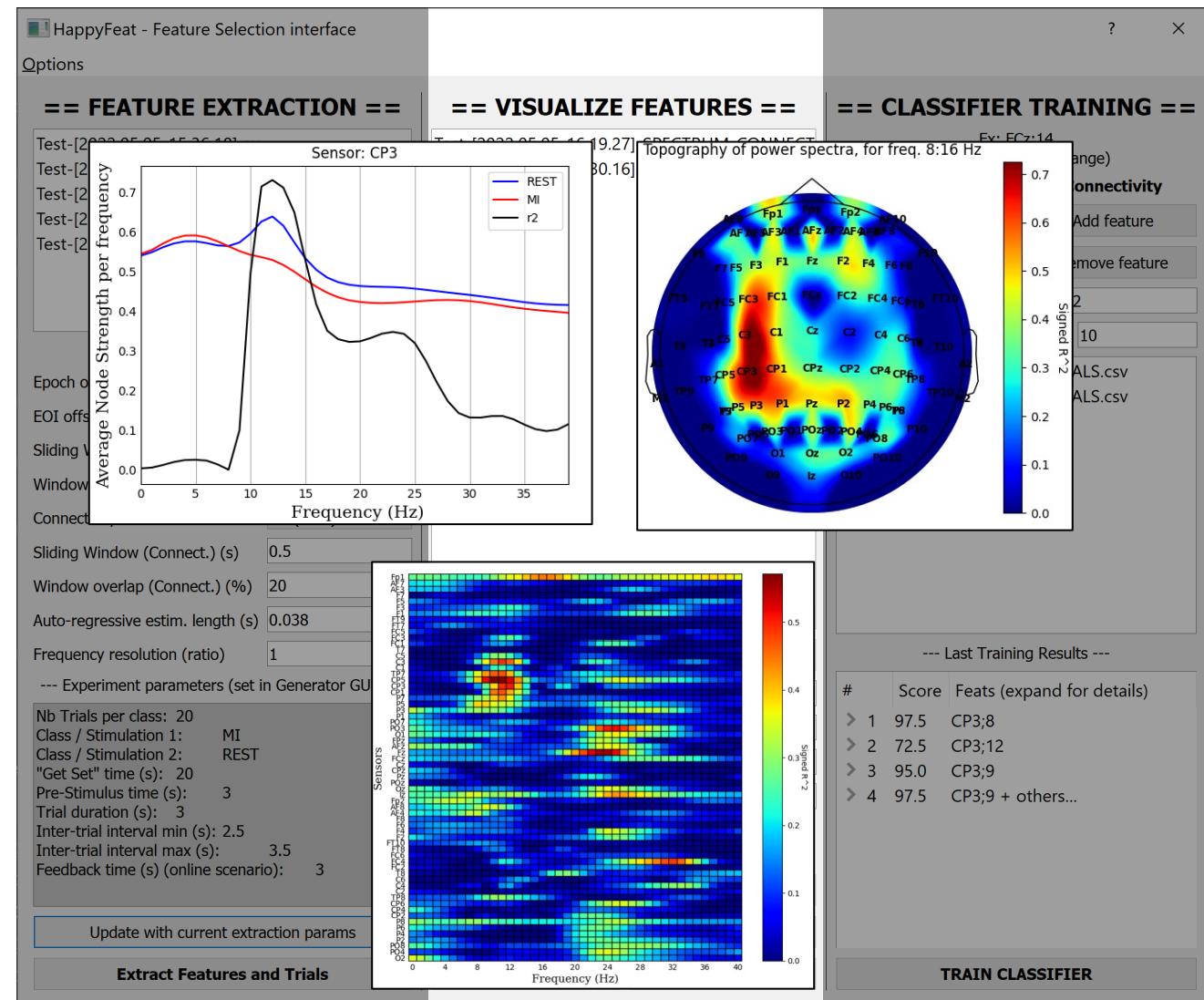


1. Workspace creation/selection

2. Feature Extraction

3. Analysis & Feature Selection

- “Extracted” files (runs) appear in the central panel.
- Select which runs to analyze...
- And use the different visualization tools to select adequate features.
- ... or use the “AutoFeat” mechanism to automatically select a set of (channel/frequency) pairs with the highest R²

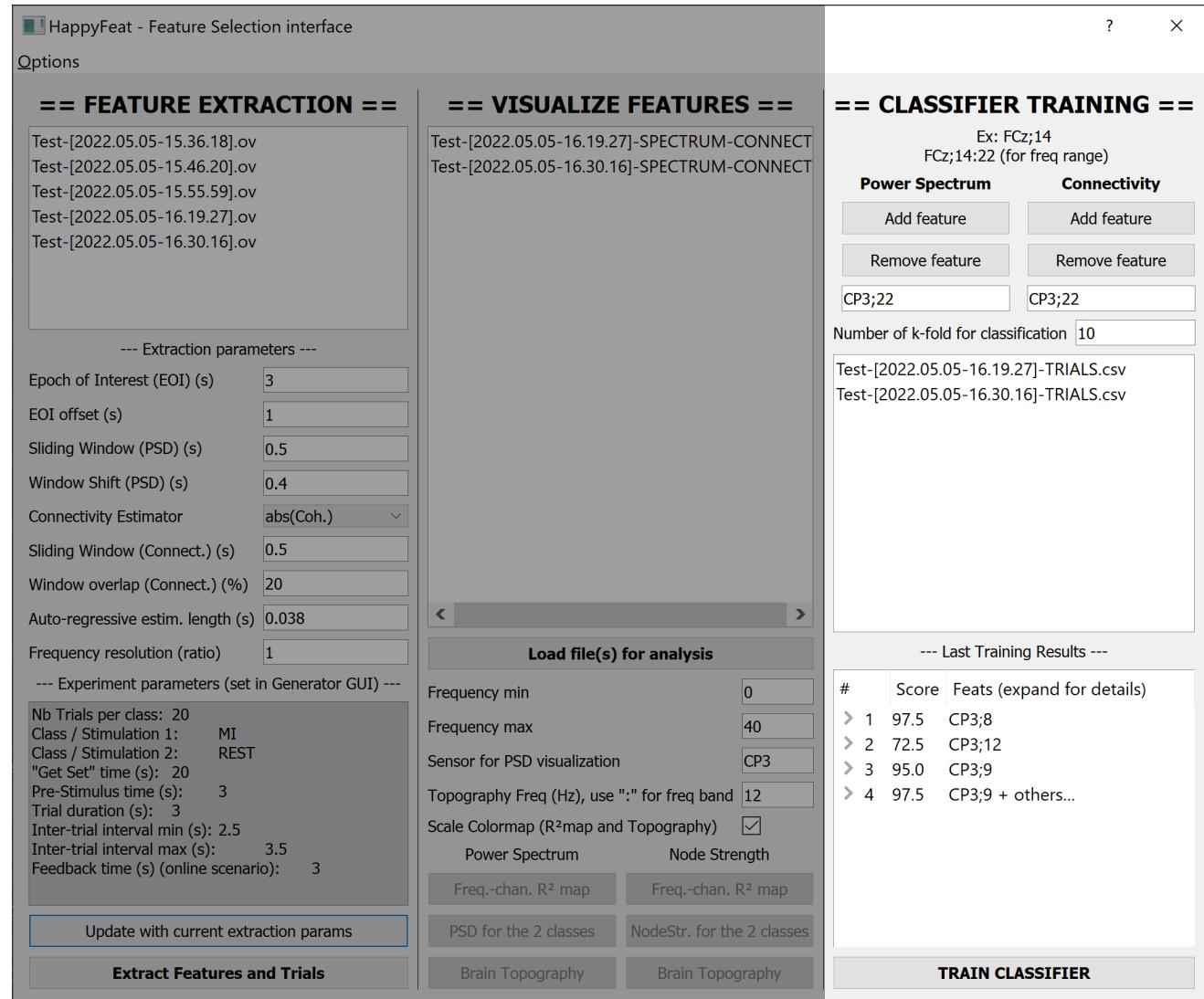


HappyFeat - Typical usecase



1. Workspace creation/selection
2. Feature Extraction
3. Analysis & Feature Selection
4. Classifier training

- “Extracted” files (runs) also appear in the right panel.
- Select one or multiple runs...
... their trials will all be used for training.
- Select one or multiple feature(s) of interest
(channel;frequency)
- Click the button!

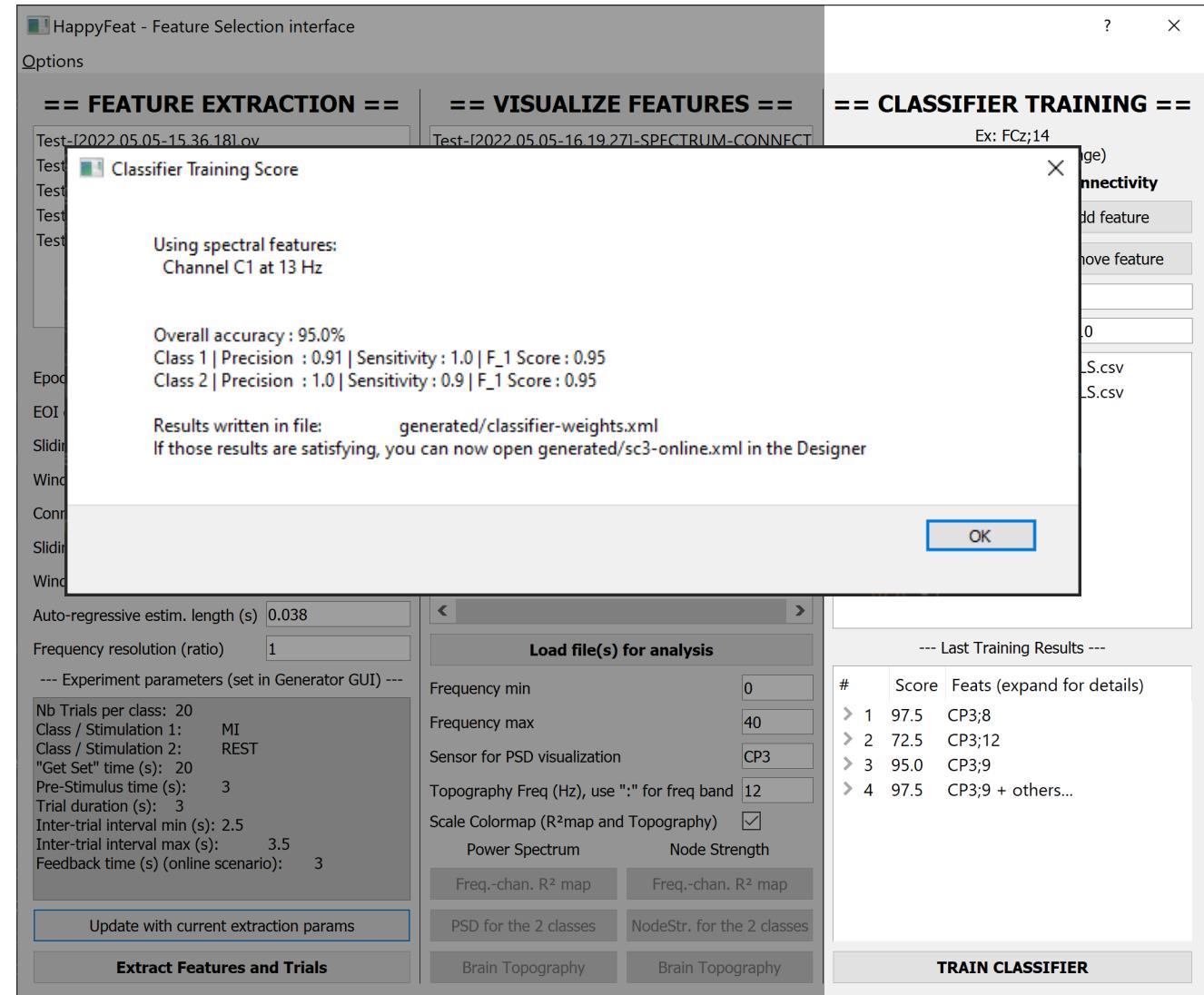


HappyFeat - Typical usecase



1. Workspace creation/selection
2. Feature Extraction
3. Analysis & Feature Selection
4. Classifier training

- “Extracted” files (runs) also appear in the right panel.
 - Select one or multiple runs...
... their trials will all be used for training.
 - Select one or multiple feature(s) of interest
(channel;frequency)
 - Click the button!
- ➔ A training accuracy is provided
- ➔ All previous training results are available
(displayed in the GUI, and classif. weights in
the workspace)



HappyFeat - Typical usecase

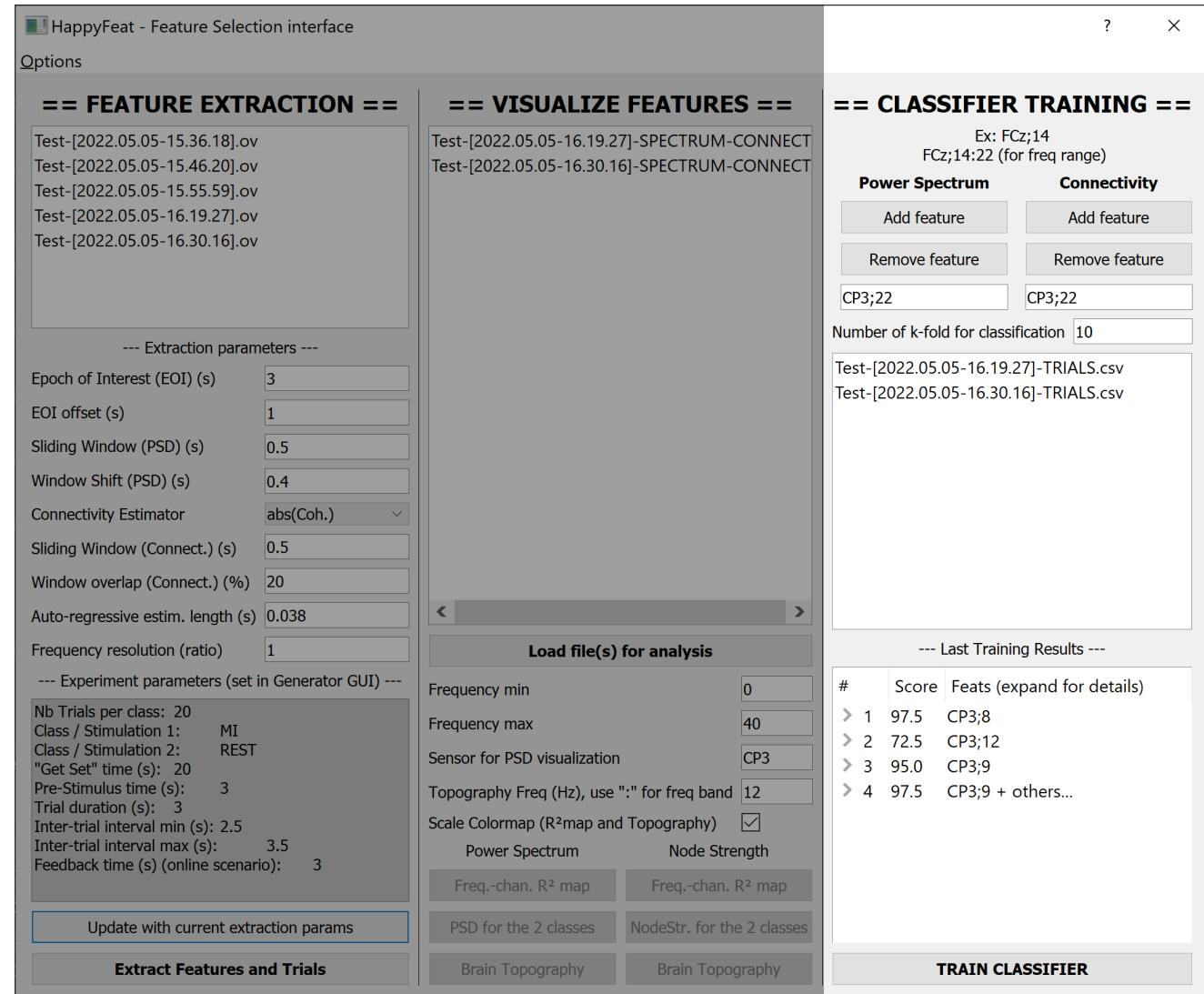


1. Workspace creation/selection
2. Feature Extraction
3. Analysis & Feature Selection
4. Classifier training
5. Online classification

- A satisfactory set of classification features has been found!

→ Generate a **ready-to-use**
OpenViBE (soon TimeFlux) scenario

- trained classifier
- selected classif.features



1. Context

- What is BCI? What are we looking for?
- Feature-based classification vs. BCI variabilities

2. HappyFeat

- Key points
- Example through typical use-case

3. Perspectives & conclusion

- **Current limitations**
 - Only one type of classification algo. proposed (LDA)
 - 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 + ...)

- Future works
 - Fully autonomous 100% Python version, without OpenViBE (in progress: *T. Venot*)
 - 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...)
 - + Research oriented app → integrating NERV's works on graphs

- **Future works**
 - *2024 internship in progress:* allowing other BCI softwares: **Timeflux**
 - Slower... (no C++ optimizations!)
 - + More portable, all types of platforms
 - + TimeFlux also 100% Python, easier communication between softwares
 - **General GUI improvements**
 - « Click & play » philosophy for feature selection

Full list of dependencies:

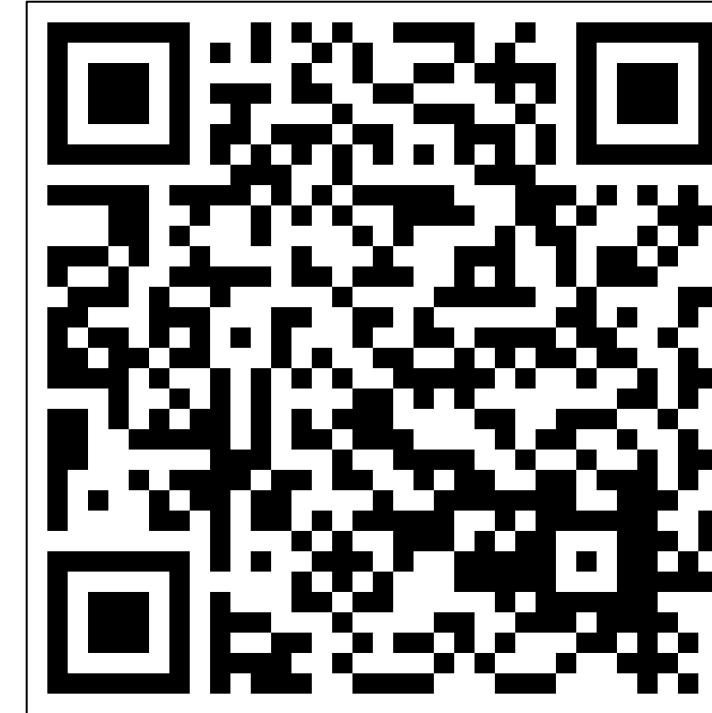
- **Python 3.9**
 - shutils>=0.1.0
 - Mne==1.5.1
 - Numpy==1.26.4
 - Pandas==2.2.1
 - PySide2==5.15.2.1
 - statsmodels>=0.13.1
 - scipy>=1.7.1
 - spectrum>=0.8.0
 - Matplotlib==3.8.4
- **OpenViBE v3.6.0**

- Available online:
 - github: <https://github.com/Inria-NERV/happyFeat>
 - PyPI: <https://pypi.org/project/happyfeat/>
 - documentation: <https://happyfeat.readthedocs.io/>
- To be continued...
 - + + flexibility (pipeline options, mixing metrics...)
 - + + network metrics (based on connectivity) (and associated viz. tools)
 - + + classification algorithms options
 - Fully autonomous Python version, “sandbox” for research purposes
 - Support for other BCI SW (starting with TimeFlux)
 - GUI & general interactivity improvement

Stay tuned!



github.com/Inria-NERV/happyFeat



<https://doi.org/10.1016/j.simpa.2023.100610>



Thanks for your attention!



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

Thanks for your attention! Any questions?



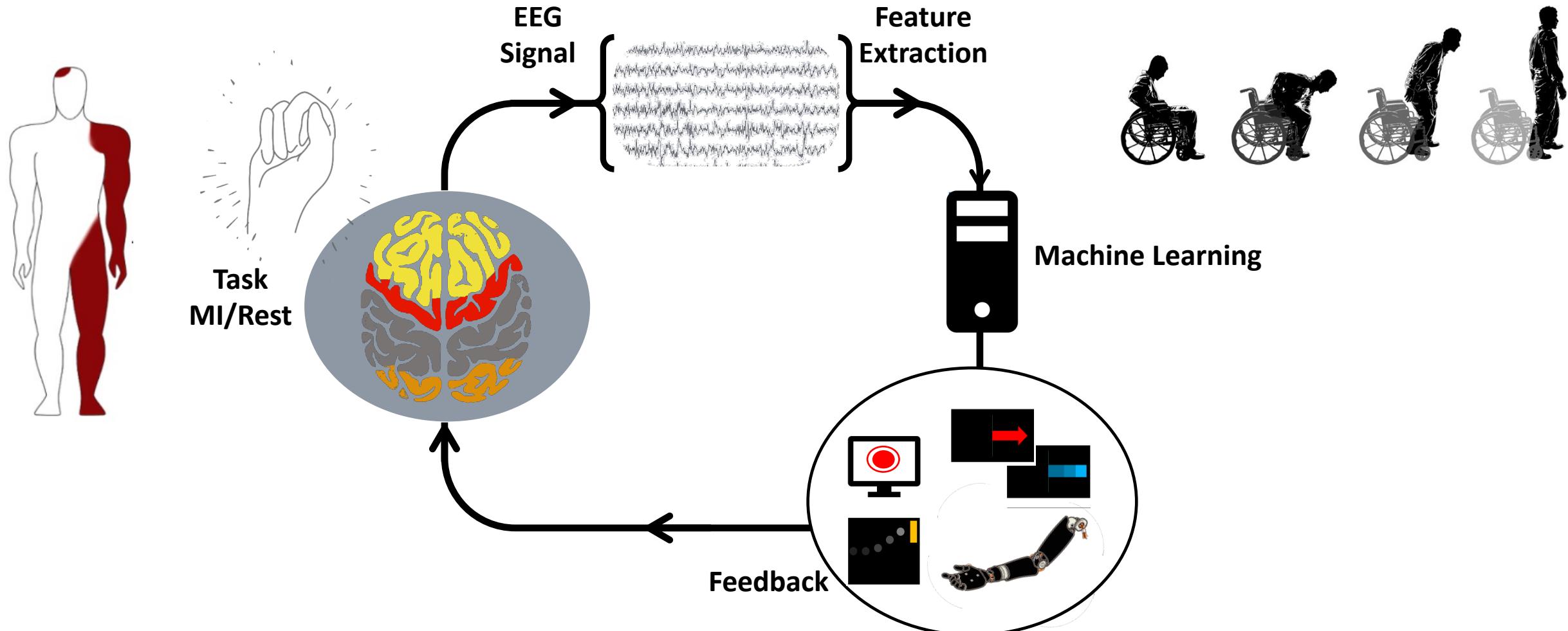
NERV BCI Journal Club - Dec 13th 2023

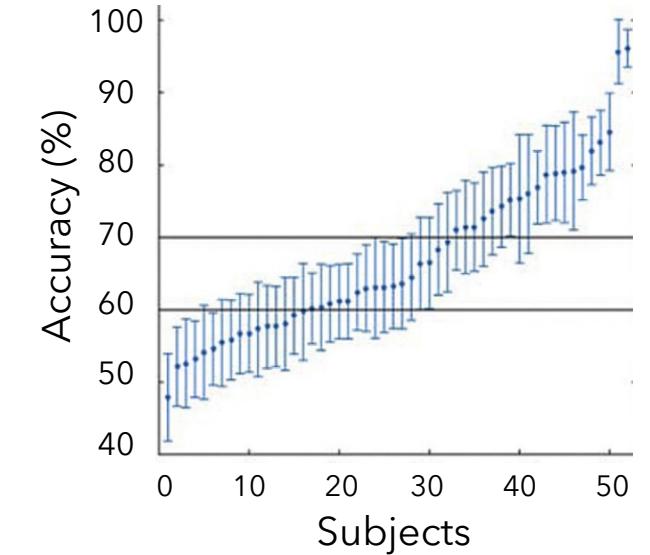
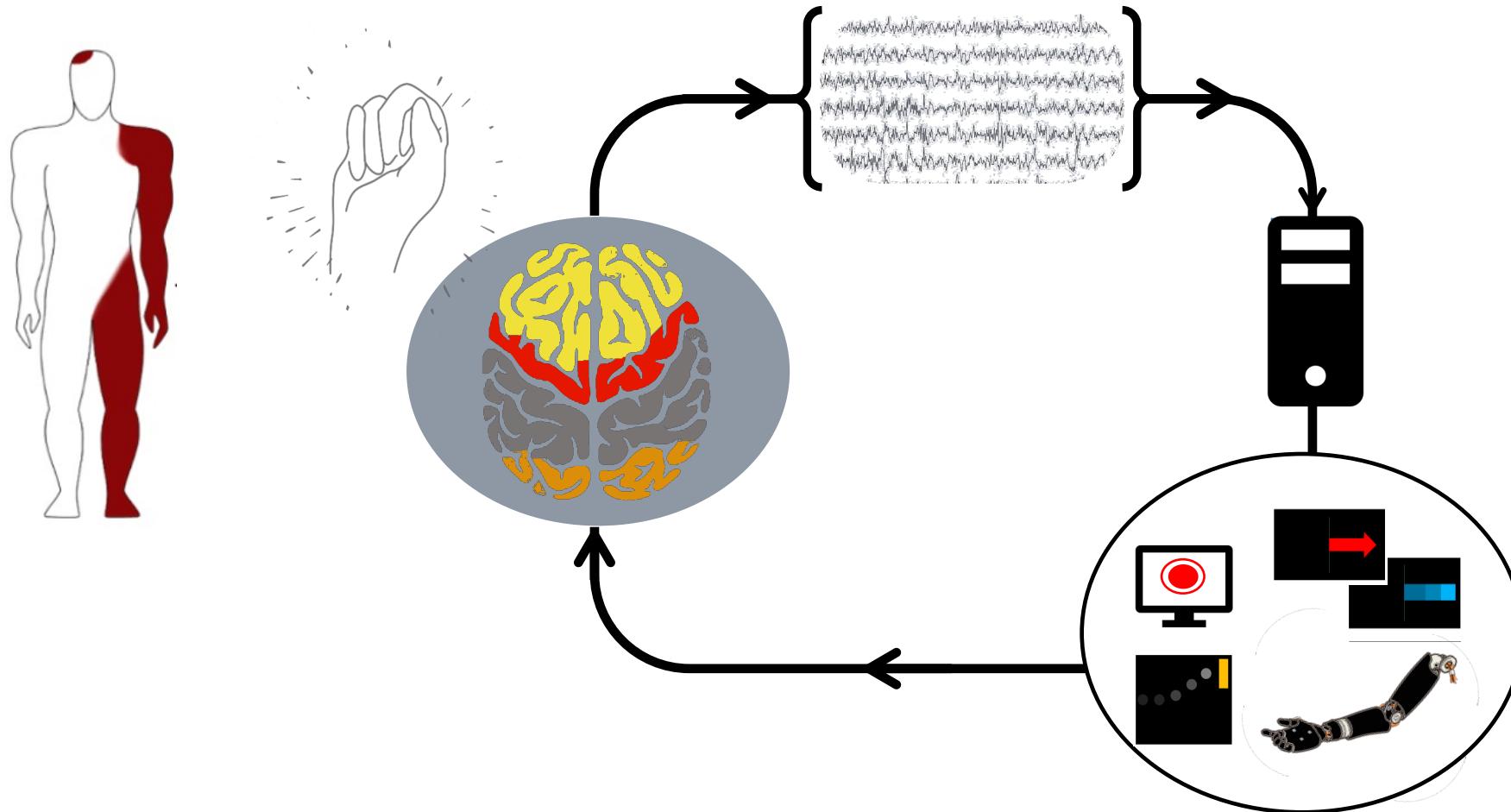
HappyFeat: an interactive and efficient BCI framework for clinical applications

Additional material

Arthur Desbois
Tristan Venot, Fabrizio De Vico Fallani, Marie-Constance Corsi
Inria Paris, NERV team, Paris Brain Institute (ICM)

Context - What is a “BCI”?





Adapted from (Ahn & Jun, 2015)

Current BCIs fail to detect the mental intentions in ~30% of users – **BCI inefficiency**

- **Sources of variabilities**

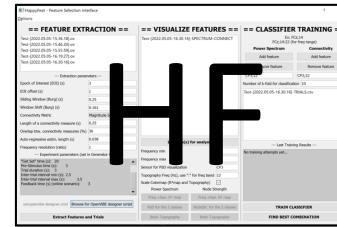
- Data acquisition (eg EEG sensors impedances)
- Tiredness
- Brain signatures (eg spatial shifts)

- ➔ Need to reduce the calibration time
- ➔ Need to identify the features that best capture the patients' intent

HappyFeat is a software aiming to simplify the use of BCI pipelines
in clinical settings by non-expert users by
assisting the extraction and the selection of the classification features.

1. Workspace creation/selection

- Create a new environment (file tree, configuration, data structures) for the experiment/analysis
- ... or select an already existing one.
- All user settings & manipulations are saved and available in the workspace for future usage or sharing:
 - Extraction parameters (spectral analysis, list of processed files...)
 - Training attempts (runs & features used, training accuracy...)

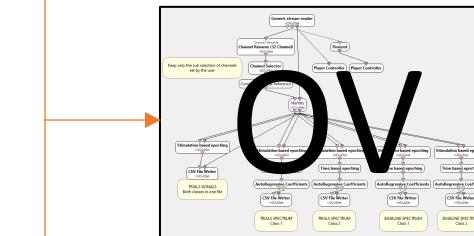
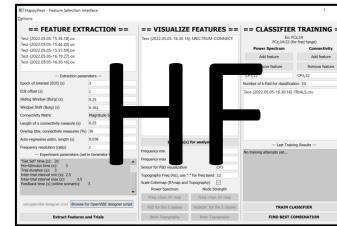


HappyFeat - How? (with openvibe)

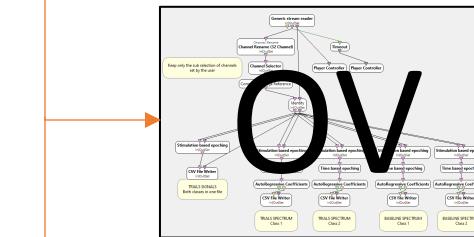


1. Workspace creation/selection
2. MI Pipeline / “Feature type” selection

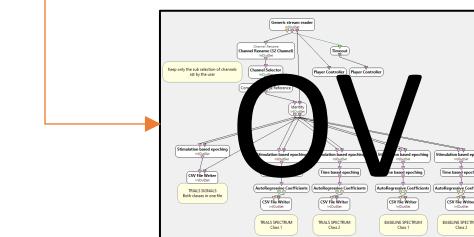
- Selecting btw. multiple “template” scenarios depending on the use case (power spectrum, connectivity type...)
- Edit basic/common parameters (acquisition, extraction, training...)



Extraction scenario



Training scenario



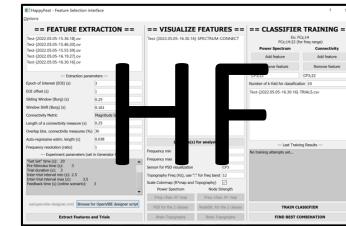
Online classif.
scenario

HappyFeat - How? (with openvibe)

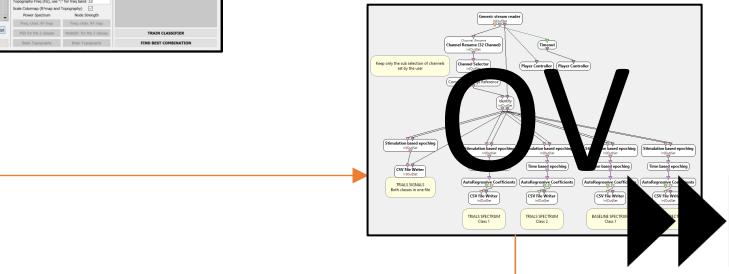


1. Workspace creation/selection
2. MI Pipeline / “Feature type” selection
3. Feature Extraction

- 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.



Extraction scenario



Signal 1
.ov

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

Signal 2
.ov

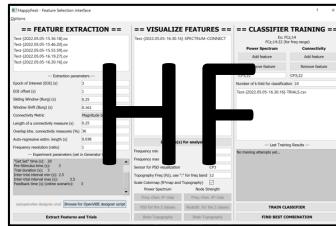
- ➔ Spectral Power (class 1) CSV
- ➔ Spectral Power (class 2) CSV
- ➔ Trials CSV
- ➔ Metadata CSV

...

HappyFeat - How? (with openvibe)

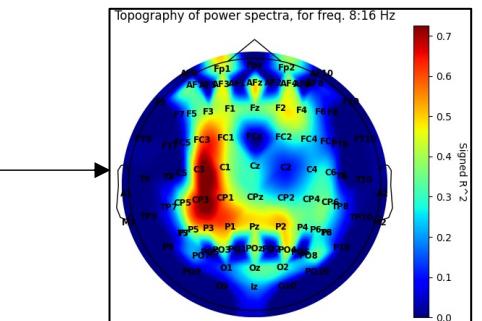
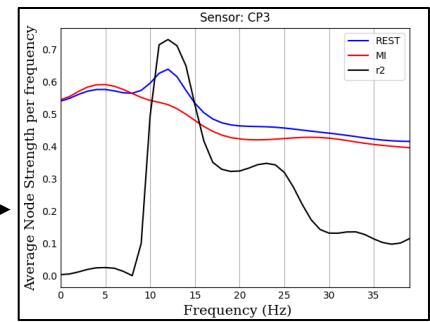
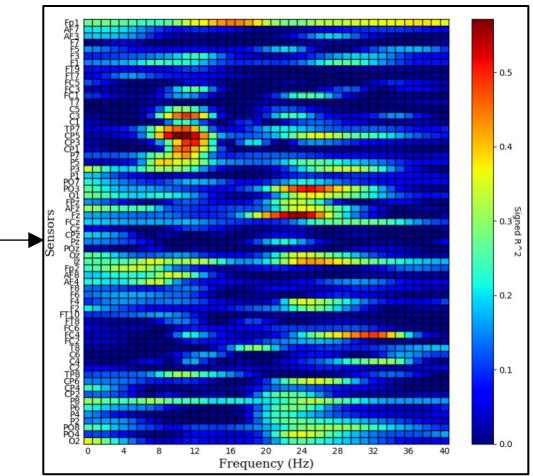
1. Workspace creation/selection
2. MI Pipeline / “Feature type” selection
3. Feature Extraction
4. Analysis, Feature Selection

- 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)



Signal 1
(.ov)

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



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

HappyFeat - How? (with openvibe)



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

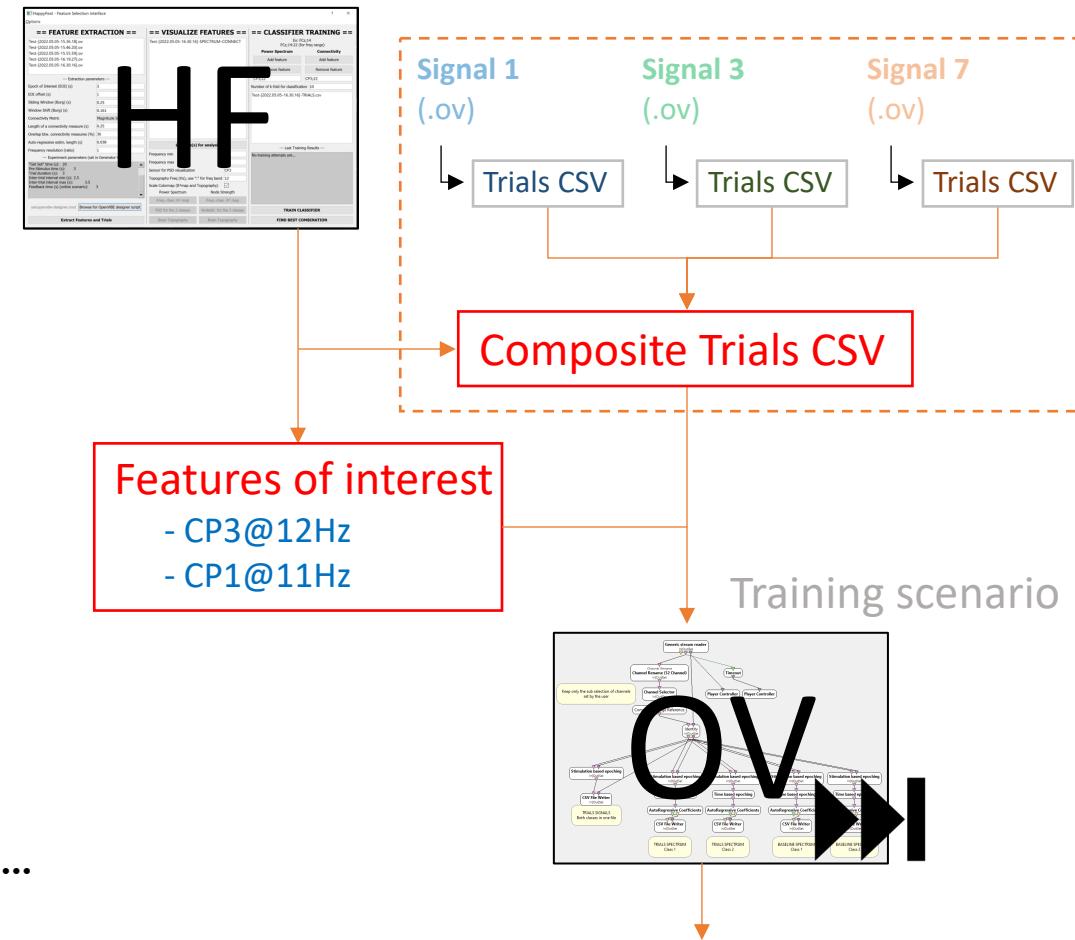
- Set one or more **Features of Interest** (Electrode/Freq.)
- **Select file(s)** with which you want to train your classifier
If > 1 file : their trials are automatically concatenated
- Run the **Classifier Training scenario**
(Auto. generated in step 1 + auto. edited with FOIs)
→ Classification **ACCURACY**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