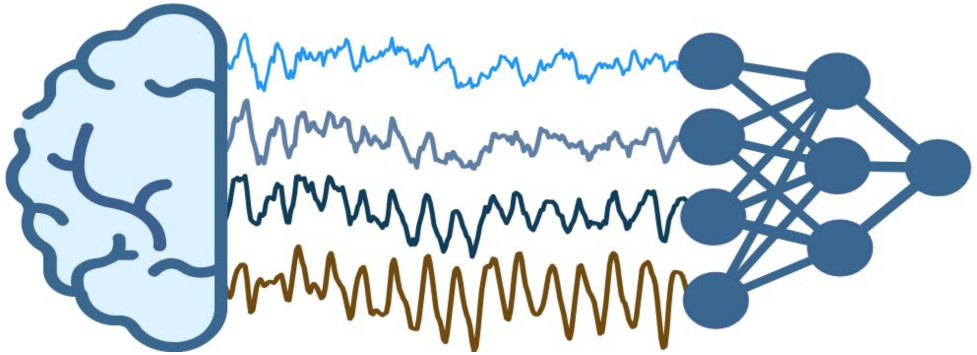
BRAINDECODE



DEEP LEARNING FOR EEG DECODING

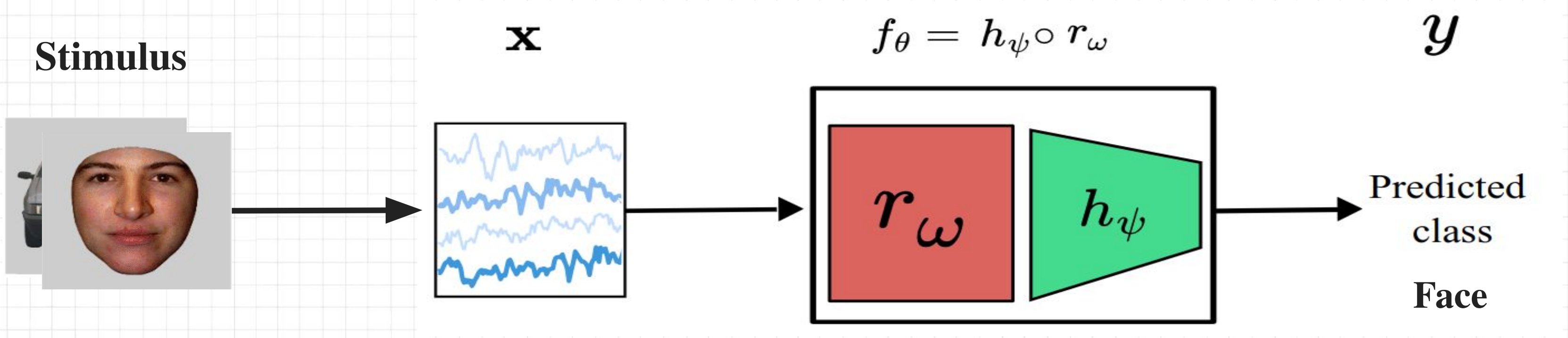
Workshop 2: Designing Brain-Computer Interfaces, from theory to real-life scenarios

Bruno Aristimunha

https://www.linkedin.com/in/bruaristimunha/
https://github.com/bruAristimunha

September 9 — 9th Graz BCI Conference 2024

What is brain decoding?



Brain decoding is translating recorded neural activity into its originating stimulus or behavior, KING, R. et al. (2020).

Small overview

- Basic data preprocessing for brain decoding.
- **24 Models** for **classification and regression.**
- **18 Data augmentations** implementations.
- **24 tutorials** teaching everyone the fundamental lego pieces to use.
- Datasets with MOABB datasets (+50 datasets), Sleep stage dataset, and more BIG datasets (Teras of data).

Traffic Stats

We are the largest library for EEG decoding and deep learning!

- 344,050 downloads (from pip).
- 4k Downloads in last month.
- 1k Downloads in the last 7 days.
- 2069 views with 282 unique visitors in the Github.

Braindecode goals!

For ***neuroscientists who want to work with deep learning*** and ***deep learning researchers who want to work with neurophysiological*** data.

For neuroscientists or more applied users: it is fully compatible with Scikit-learn.

Data Supports

Braindecode is compatible with several file formats thanks to MNE and NumPy arrays.

You can use your raw or epoched data and build a simple script to decode a domain-specific task.

MEG + EEG ANALYSIS & VISUALIZATION

Check one tutorial about this

For the deep learning experts

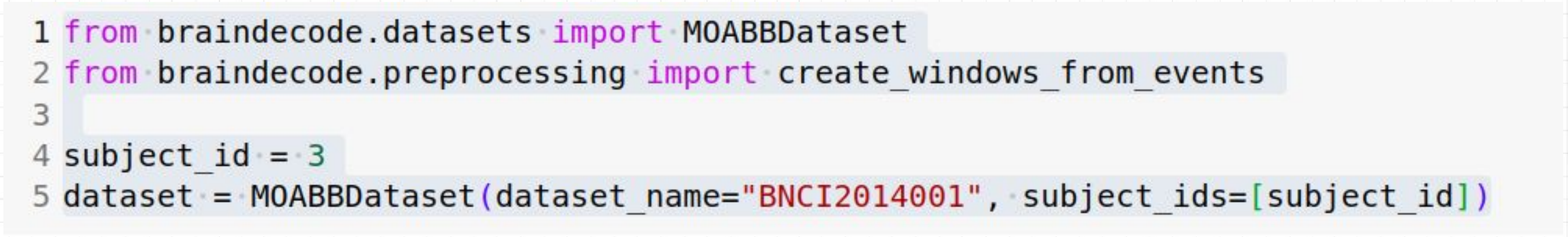
All our models are implemented in PyTorch, with a strong convention for model layers and the same variable names for things related to the EEG signal.

What I mean here is we have **solid standardization**, **code with many tests**, and **good modularization**!

Check one PyTorch tutorial

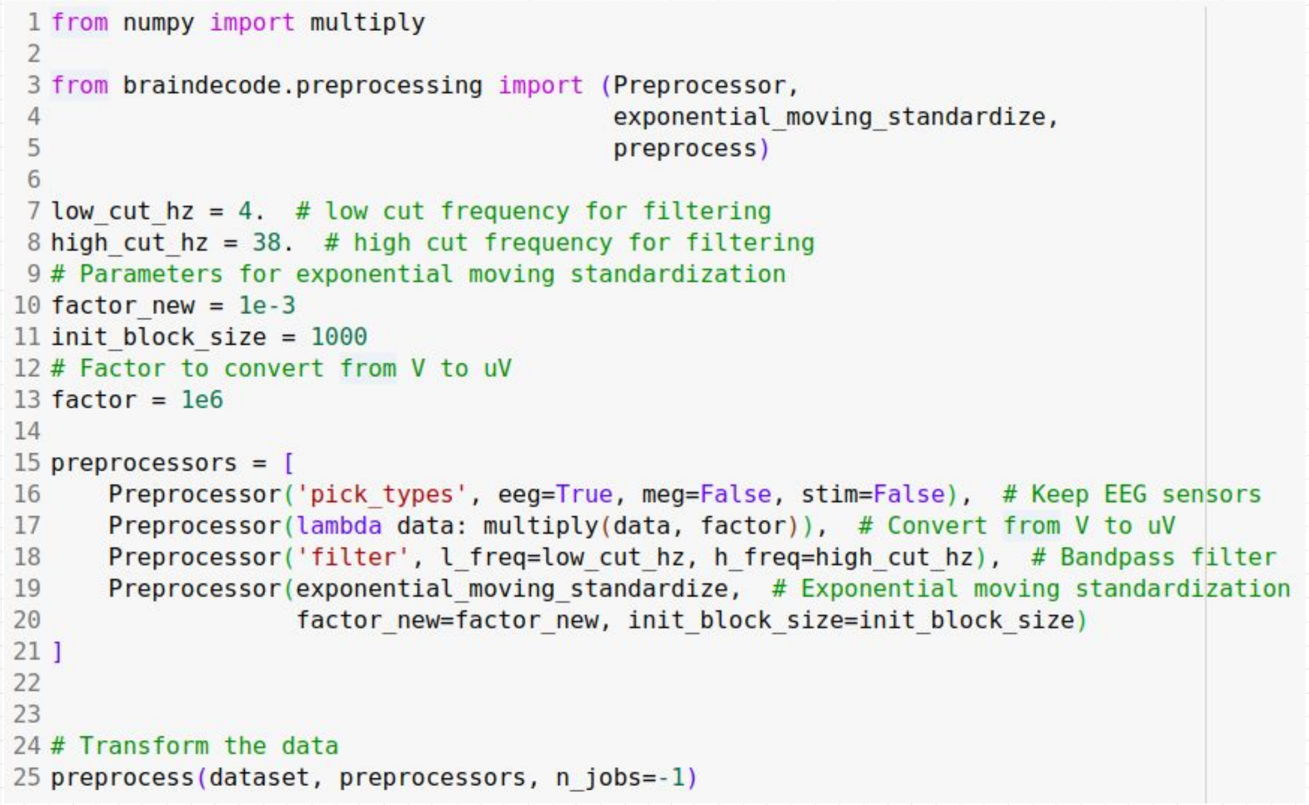
Braindecode: step by step

1) You define an dataset object!



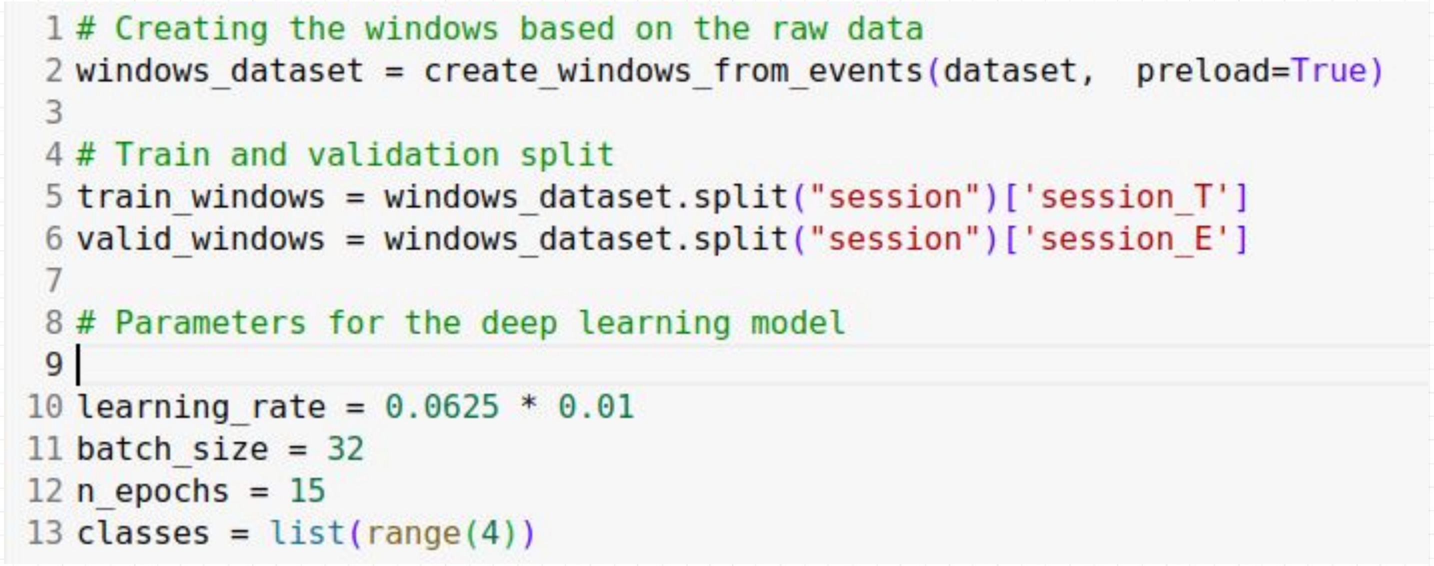
```python
from braindecode.datasets import MOABBDataset
from braindecode.preprocessing import create_windows_from_events

subject_id = 3
dataset = MOABBDataset(dataset_name="BNCI2014001", subject_ids=[subject_id])
```

Pre-processing on the data



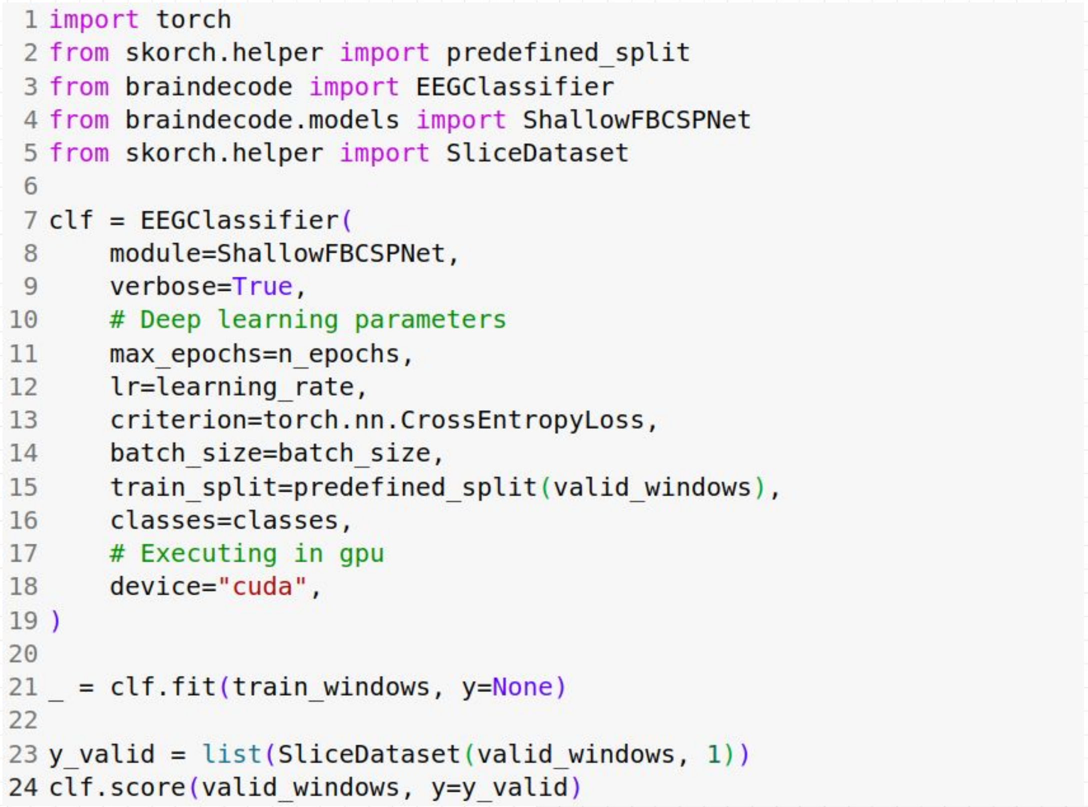
```python
from numpy import multiply

from braindecode.preprocessing import (Preprocessor,
                                       exponential_moving_standardize,
                                       preprocess)

low_cut_hz = 4.  # low cut frequency for filtering
high_cut_hz = 38.  # high cut frequency for filtering
# Parameters for exponential moving standardization
factor_new = 1e-3
init_block_size = 1000
# Factor to convert from V to uV
factor = 1e6

preprocessors = [
    Preprocessor('pick_types', eeg=True, meg=False, stim=False),  # Keep EEG sensors
    Preprocessor(lambda data: multiply(data, factor)),  # Convert from V to uV
    Preprocessor('filter', l_freq=low_cut_hz, h_freq=high_cut_hz),  # Bandpass filter
    Preprocessor(exponential_moving_standardize,  # Exponential moving standardization
                 factor_new=factor_new, init_block_size=init_block_size)
]


# Transform the data
preprocess(dataset, preprocessors, n_jobs=-1)
```

Windows, train-valid, parameters



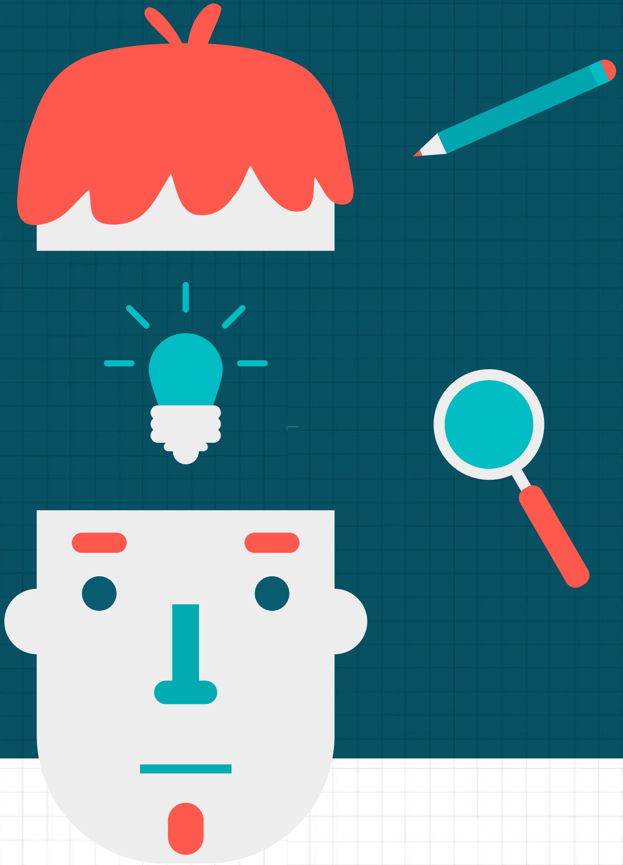
```python
# Creating the windows based on the raw data
windows_dataset = create_windows_from_events(dataset,  preload=True)

# Train and validation split
train_windows = windows_dataset.split("session")['session_T']
valid_windows = windows_dataset.split("session")['session_E']

# Parameters for the deep learning model

learning_rate = 0.0625 * 0.01
batch_size = 32
n_epochs = 15
classes = list(range(4))
```

Training and evaluation the model



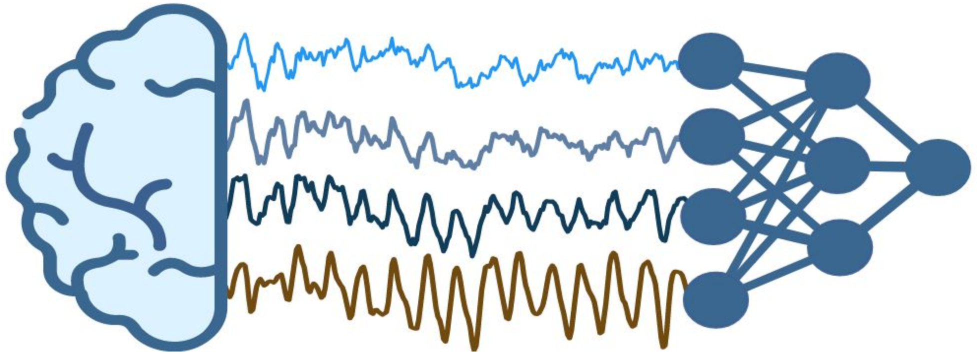
```python
import torch
from skorch.helper import predefined_split
from braindecode import EEGClassifier
from braindecode.models import ShallowFBCSPNet
from skorch.helper import SliceDataset

clf = EEGClassifier(
    module=ShallowFBCSPNet,
    verbose=True,
    # Deep learning parameters
    max_epochs=n_epochs,
    lr=learning_rate,
    criterion=torch.nn.CrossEntropyLoss,
    batch_size=batch_size,
    train_split=predefined_split(valid_windows),
    classes=classes,
    # Executing in gpu
    device="cuda",
)

_ = clf.fit(train_windows, y=None)

y_valid = list(SliceDataset(valid_windows, 1))
clf.score(valid_windows, y=y_valid)
```

Getting-started tutorials



Bruno Aristimunha
b.aristimunha@gmail.com

BRAINDECODE



www.braindecode.org

We are open to contribution at Github!
https://github.com/braindecode/braindecode