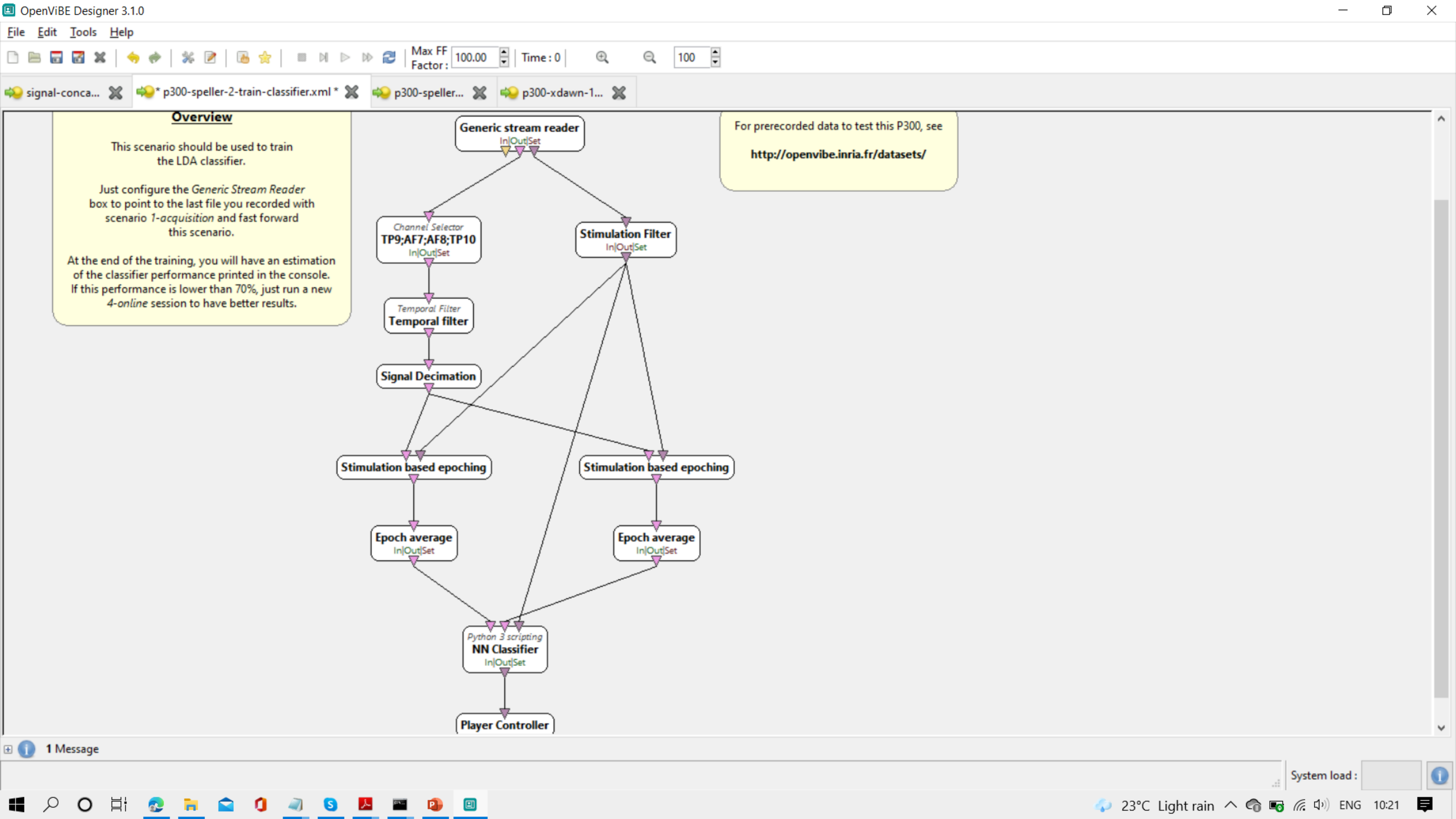
Goal

Get P300 Speller working with CNN classifier and state its advantages



Overview

This scenario should be used to train the LDA classifier.

Just configure the *Generic Stream Reader* box to point to the last file you recorded with scenario *1-acquisition* and fast forward this scenario.

At the end of the training, you will have an estimation of the classifier performance printed in the console. If this performance is lower than 70%, just run a new *4-online* session to have better results.

For prerecorded data to test this P300, see

http://openvibe.inria.fr/datasets/

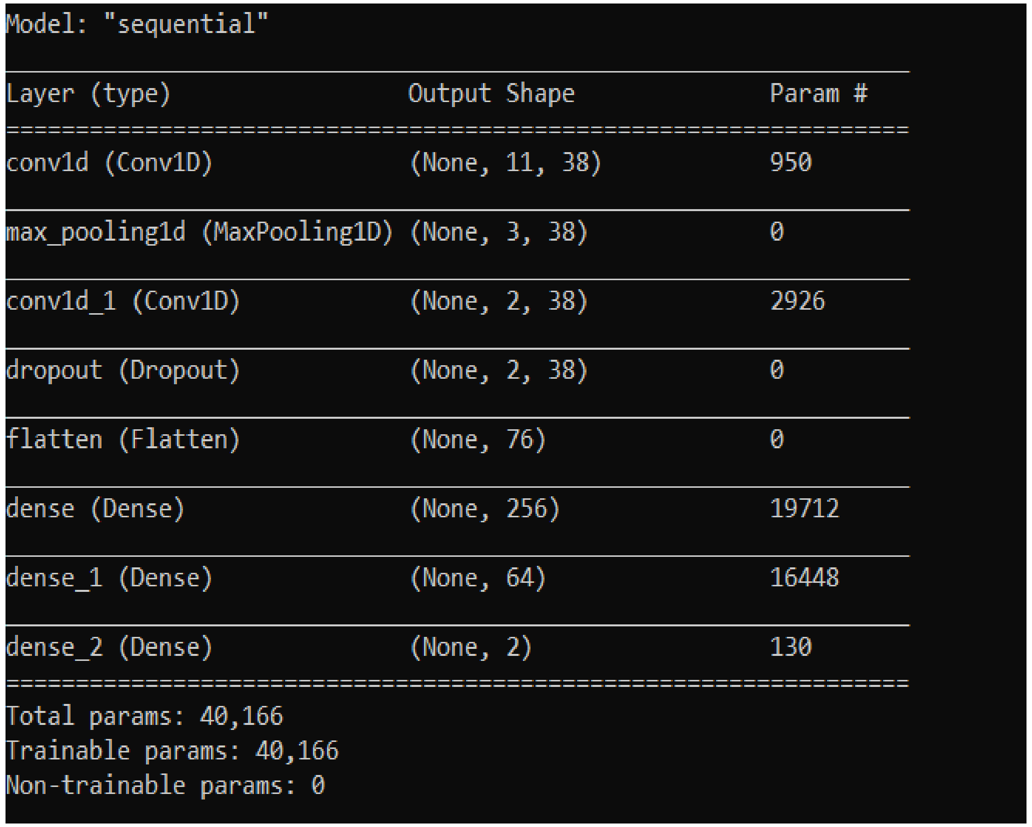
CNN Model

https://cs231n.github.io/convolutional-networks/

```
# Create the model
model=Sequential()
model.add(Conv1D(filters=38, kernel_size=6, padding='valid', activation='relu', strides=3, input_shape=(38, 4)))
model.add(MaxPooling1D(pool_size=3))
model.add(Conv1D(filters=38, kernel_size=2, padding='valid', activation='relu', strides=1))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(2, activation='softmax'))

model.summary()

# Compile the model
sgd = SGD(lr=0.01, momentum=0.7, nesterov=True)     # Higher values get steep loss curves ..
model.compile(loss=sparse_categorical_crossentropy, optimizer=sgd, metrics=['accuracy'])
```

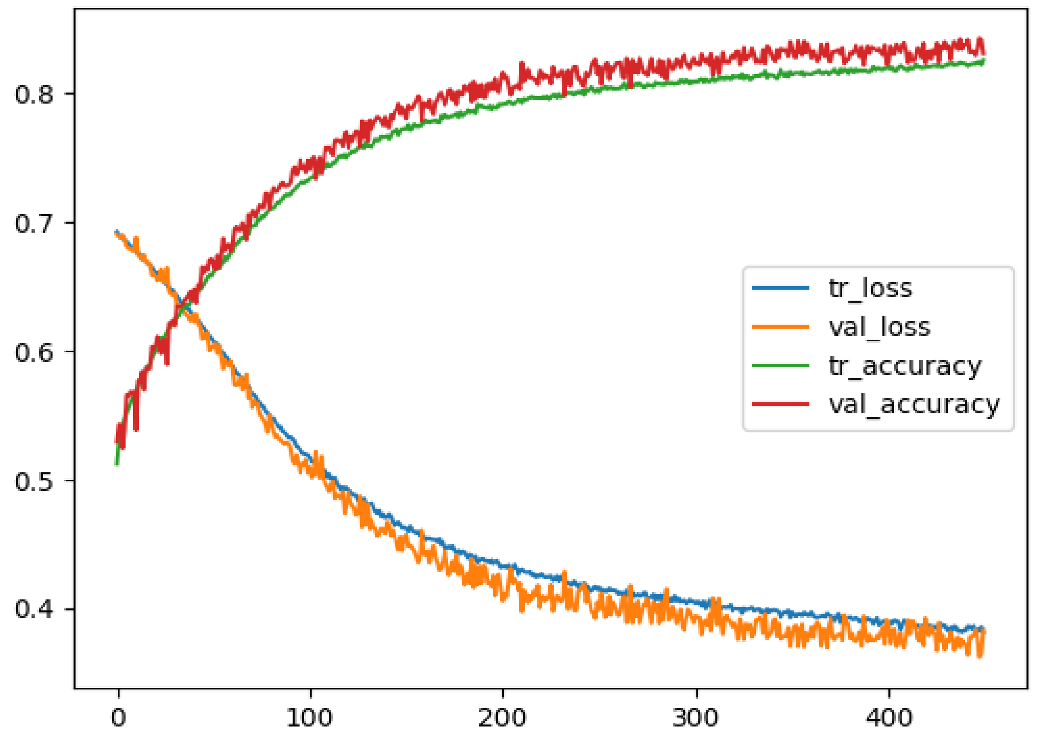


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 11, 38)            950
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 3, 38)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2, 38)             2926
_________________________________________________________________
dropout (Dropout)            (None, 2, 38)             0
_________________________________________________________________
flatten (Flatten)            (None, 76)                0
_________________________________________________________________
dense (Dense)                (None, 256)               19712
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 40,166
Trainable params: 40,166
Non-trainable params: 0
```

```
hist = model.fit(input, Y_train, batch_size=32, epochs=450, verbose=1, validation_data=(testinput, Y_test), steps_per_epoch=None)
```

Params = (channels * kernel * filters) + filters (number of bias)

950 = 4 * 6 * 38 + 38



Changes in Confusion Matrix

Ideal Matrix $\begin{bmatrix} 1200 & 0 \\ 0 & 240 \end{bmatrix}$ Online prediction for 20 characters, 6 repetitions

120 Training

Characters $\begin{bmatrix} 1061 & 139 \\ 219 & 21 \end{bmatrix}$ Online prediction for 20 characters, 6 repetitions

240 Training

Characters $\begin{bmatrix} 968 & 232 \\ 197 & 43 \end{bmatrix}$ Online prediction for 20 characters, 6 repetitions

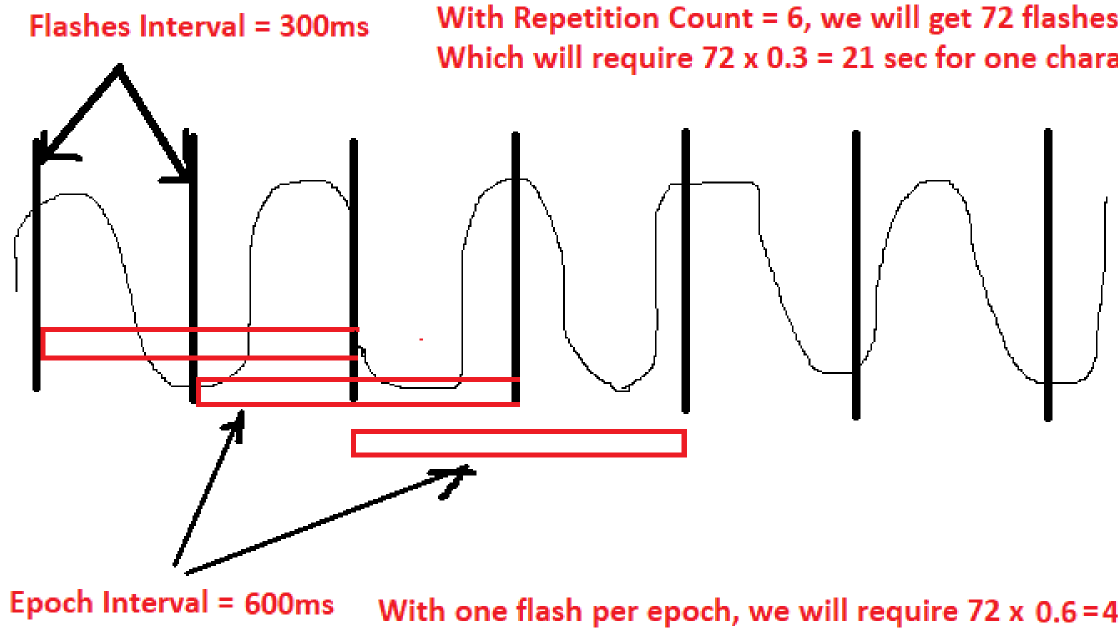
360 Training

Characters $\begin{bmatrix} 856 & 344 \\ 166 & 74 \end{bmatrix}$ Online prediction for 20 characters, 6 repetitions

480 Training

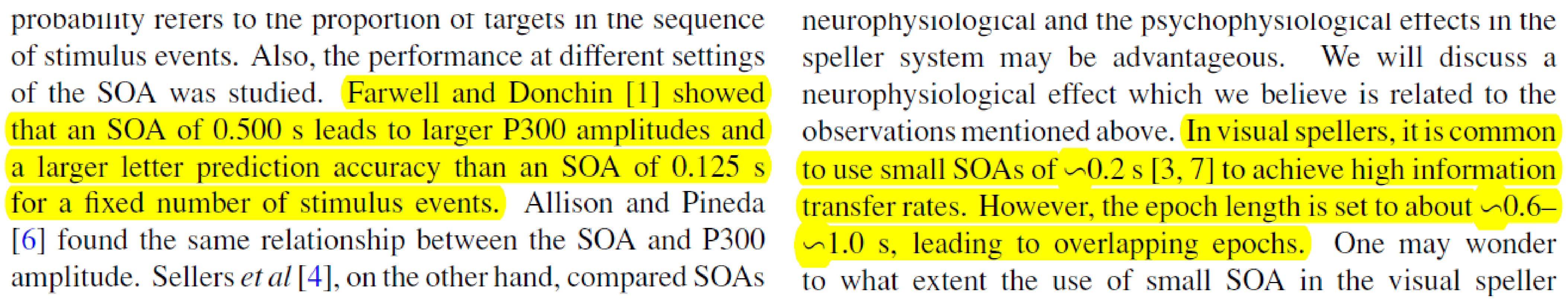
Characters $\begin{bmatrix} 629 & 571 \\ 120 & 120 \end{bmatrix}$ Online prediction for 20 characters, 6 repetitions

Overlapping Epochs



With Repetition Count = 6, we will get 72 flashes for one character.
Which will require 72 x 0.3 = 21 sec for one character prediction.

With one flash per epoch, we will require 72 x 0.6 = 43 sec for one charcter prediction

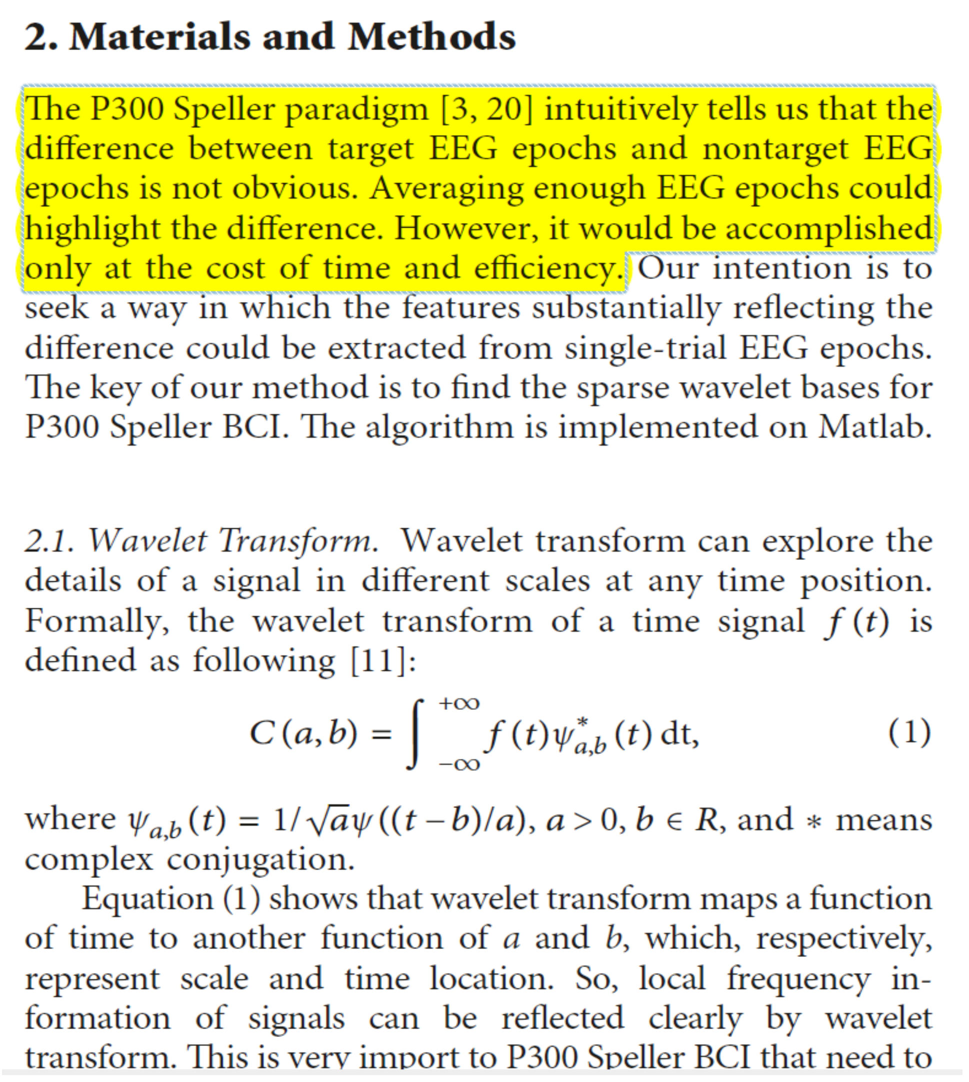


probability refers to the proportion of targets in the sequence of stimulus events. Also, the performance at different settings of the SOA was studied. Farwell and Donchin [1] showed that an SOA of 0.500 s leads to larger P300 amplitudes and a larger letter prediction accuracy than an SOA of 0.125 s for a fixed number of stimulus events. Allison and Pineda [6] found the same relationship between the SOA and P300 amplitude. Sellers *et al* [4], on the other hand, compared SOAs

neurophysiological and the psychophysiological effects in the speller system may be advantageous. We will discuss a neurophysiological effect which we believe is related to the observations mentioned above. In visual spellers, it is common to use small SOAs of ∽0.2 s [3, 7] to achieve high information transfer rates. However, the epoch length is set to about ∽0.6–∽1.0 s, leading to overlapping epochs. One may wonder to what extent the use of small SOA in the visual speller

Epoch average

[Documentation: Home (inria.fr)]



2. Materials and Methods

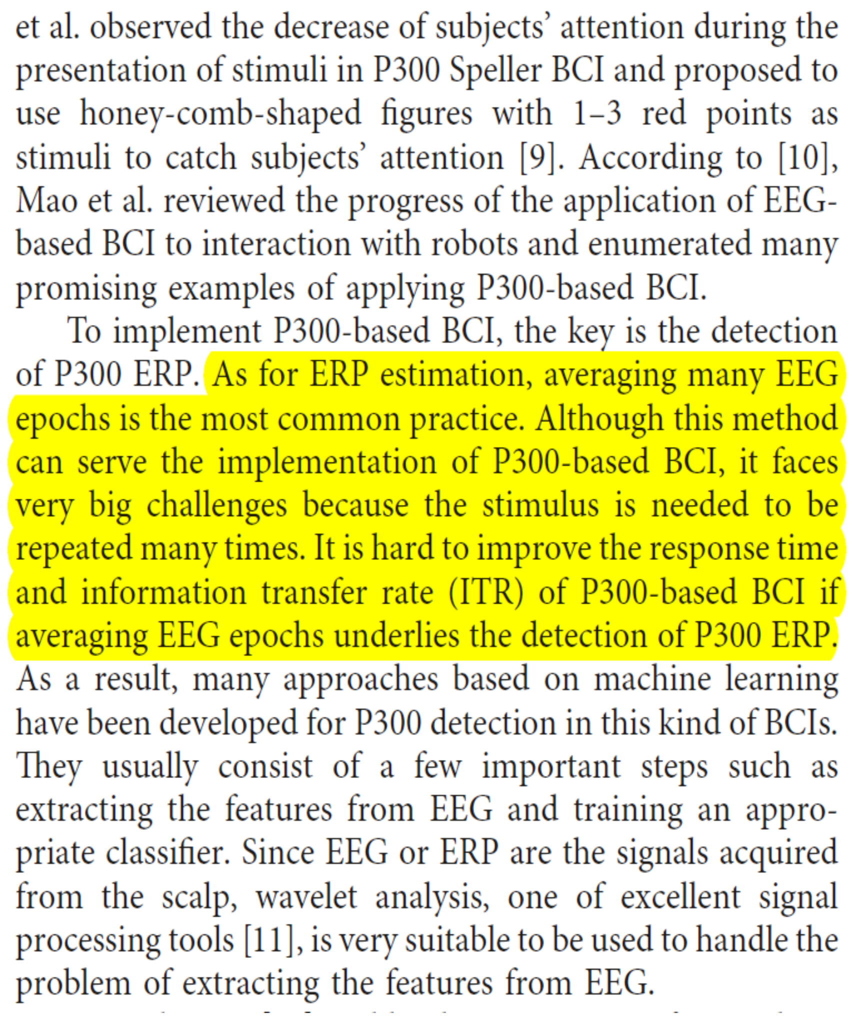
The P300 Speller paradigm [3, 20] intuitively tells us that the difference between target EEG epochs and nontarget EEG epochs is not obvious. Averaging enough EEG epochs could highlight the difference. However, it would be accomplished only at the cost of time and efficiency. Our intention is to seek a way in which the features substantially reflecting the difference could be extracted from single-trial EEG epochs. The key of our method is to find the sparse wavelet bases for P300 Speller BCI. The algorithm is implemented on Matlab.

2.1. Wavelet Transform. Wavelet transform can explore the details of a signal in different scales at any time position. Formally, the wavelet transform of a time signal $f(t)$ is defined as following [11]:

$$C(a,b) = \int_{-\infty}^{+\infty} f(t)\psi_{a,b}^{*}(t)\,\mathrm{d}t, \qquad (1)$$

where $\psi_{a,b}(t) = 1/\sqrt{a}\psi((t-b)/a)$, $a > 0$, $b \in R$, and $*$ means complex conjugation.

Equation (1) shows that wavelet transform maps a function of time to another function of a and b, which, respectively, represent scale and time location. So, local frequency information of signals can be reflected clearly by wavelet transform. This is very import to P300 Speller BCI that need to



et al. observed the decrease of subjects' attention during the presentation of stimuli in P300 Speller BCI and proposed to use honey-comb-shaped figures with 1–3 red points as stimuli to catch subjects' attention [9]. According to [10], Mao et al. reviewed the progress of the application of EEG-based BCI to interaction with robots and enumerated many promising examples of applying P300-based BCI.

To implement P300-based BCI, the key is the detection of P300 ERP. As for ERP estimation, averaging many EEG epochs is the most common practice. Although this method can serve the implementation of P300-based BCI, it faces very big challenges because the stimulus is needed to be repeated many times. It is hard to improve the response time and information transfer rate (ITR) of P300-based BCI if averaging EEG epochs underlies the detection of P300 ERP. As a result, many approaches based on machine learning have been developed for P300 detection in this kind of BCIs. They usually consist of a few important steps such as extracting the features from EEG and training an appropriate classifier. Since EEG or ERP are the signals acquired from the scalp, wavelet analysis, one of excellent signal processing tools [11], is very suitable to be used to handle the problem of extracting the features from EEG.

Future Work

- Classification of online flashes can be improved by keeping the interstimulus interval greater than 400ms

- Apply epoch average principle to estimation ERP

- More data with multiple subjects (more variations)