Time-adaptive Unsupervised Auditory Attention Decoding Using EEG-based Stimulus Reconstruction

Simon Geirnaert, Tom Francart, and Alexander Bertrand, *Senior Member, IEEE*

***Abstract*—The goal of auditory attention decoding (AAD) is to determine to which speaker out of multiple competing speakers a listener is attending based on the brain signals recorded via, e.g., electroencephalography (EEG). AAD algorithms are a fundamental building block of so-called neuro-steered hearing devices that would allow identifying the speaker that should be amplified based on the brain activity. A common approach is to train a subject-specific decoder that reconstructs the amplitude envelope of the attended speech signal. However, training this decoder requires a dedicated 'ground-truth' EEG recording of the subject under test, during which the attended speaker is known. Furthermore, this decoder remains fixed during operation and can thus not adapt to changing conditions and situations. Therefore, we propose an online time-adaptive unsupervised stimulus reconstruction method that continuously and automatically adapts over time when new EEG and audio data are streaming in. The adaptive decoder does not require ground-truth attention labels obtained from a training session with the end-user, and instead can be initialized with a generic subject-independent decoder or even completely random values. We propose two different implementations: a sliding window and recursive implementation, which we extensively validate based on multiple performance metrics on three independent datasets. We show that the proposed time-adaptive unsupervised decoder outperforms a time-invariant supervised decoder, representing an important step towards practically applicable AAD algorithms for neuro-steered hearing devices.**

***Index Terms*—auditory attention decoding, neuro-steered hearing device, stimulus reconstruction, time-adaptive filtering, unsupervised training, EEG**

I. Introduction

The auditory attention decoding (AAD) task consists of determining to which speaker out of multiple simultaneously talking speakers a listener wants to attend. Such AAD technology could be employed in hearing aids, cochlear implants, or other so-called 'hearables', to inform beamformers and noise suppression algorithms about which speaker to enhance and which other speakers to treat as background noise and thus to suppress. Heuristics such as look direction can be used to make an informed guess, yet such indirect metrics often fail as they do not always sufficiently correlate with the actual auditory attention. Ideally, the auditory attention can be directly decoded at the source, i.e., the brain. It has been extensively shown that the auditory attention information is encoded in brain signals [1]–[3], which can be recorded, for example, using electroencephalography (EEG). As such, EEG-based AAD technology could contribute to so-called 'neuro-steered' hearing devices [4], [5].

The predominant paradigm for AAD is stimulus reconstruction [6]. A decoder, consisting of a linear spatio-temporal filter, is applied to multi-channel EEG data of the listener to reconstruct the speech envelope of the attended speaker. By computing the correlation between this reconstructed speech envelope and the original speech envelopes of the different competing speakers, the attended speech signal and corresponding speaker can be identified [5]–[7]. However, such a stimulus decoder is traditionally trained in a supervised way, i.e., it assumes the attended speaker is known during the training phase of the decoder. Such ground-truth labels can be obtained by instructing the subject to attend to a specific speaker during a dedicated EEG recording session.

In [8], we proposed an unsupervised algorithm to train a subject-specific stimulus decoder without the need for ground-truth labels. Consequently, the first issue with the traditional stimulus reconstruction method, i.e., the need for acquiring labeled data during a dedicated training session, is resolved while approximating the performance of a fully supervised subject-specific decoder [8]. However, this unsupervised decoder is still trained in batch on a large amount of (unlabeled) training data, and then remains fixed during operation. Such fixed decoders do not adapt to long-term signal changes due to changing conditions and situations (e.g., non-stationarities in the neural activity, changing electrode-skin contact impedances, shifting or loosening electrodes). Therefore, in this paper, we modify and extend the algorithm proposed in [8] such that the decoder adapts over time in an unsupervised manner. The resulting decoder does not require a dedicated training session and can automatically adapt to new incoming non-stationary EEG data from the end-user and thus serves as one of the first practical plug-and-play AAD algorithms for neuro-steered hearing devices.

In Section II, we review the (unsupervised) stimulus recon-

This work is supported by an Aspirant Grant from the Research Foundation - Flanders (FWO) (for S. Geirnaert - 1136219N), FWO project nr. G0A4918N, the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (grant agreement No 802895), and the Flemish Government (AI Research Program). The scientific responsibility is assumed by its authors. *(Corresponding author: Simon Geirnaert.)*

S. Geirnaert and A. Bertrand are with KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics and with Leuven.AI - KU Leuven institute for AI, Kasteelpark Arenberg 10, B−3001 Leuven, Belgium (e-mail: simon.geirnaert@esat.kuleuven.be, alexander.bertrand@esat.kuleuven.be).

T. Francart and S. Geirnaert are with KU Leuven, Department of Neurosciences, Research Group ExpORL, Herestraat 49 box 721, B−3000 Leuven, Belgium and with Leuven.AI - KU Leuven institute for AI (e-mail: tom.francart@kuleuven.be).

struction algorithm for AAD. In Section III, we then present and explain the proposed time-adaptive updating schemes. These implementations are investigated and the hyperparameter choices are validated in Section IV. In Section V, we showcase how our proposed time-adaptive unsupervised stimulus reconstruction method can be used in an online context to adapt to varying conditions and situations, and compare it to the fixed (time-invariant) supervised decoder.

II. (Un)supervised stimulus reconstruction for AAD

A. Review of stimulus reconstruction

Consider a C-channel EEG signal of which the cth channel is denoted by $x_c(t)$, with t the time sample index. In the linear stimulus reconstruction paradigm, a spatio-temporal filter or decoder $d_c(l)$ is applied to this C-channel EEG signal to reconstruct the speech envelope of the attended speaker $s_{\text{a}}(t)$ [5]–[7]:

$$\hat{s}_{\text{a}}(t) = \sum_{c=1}^{C} \sum_{l=0}^{L-1} d_c(l) x_c(t+l),$$

with the channel index c ranging from 1 to C (spatial combination of C channels) and the time lag index l ranging from 0 to $L-1$ (temporal integration over L time samples). This filter is an anti-causal filter, as L *post-stimulus* time lags are used to reconstruct the attended speech envelope from the EEG signal. In the AAD problem, to identify the attended speaker, the reconstructed envelope $\hat{s}_{\text{a}}(t)$ from the EEG is compared with original speech envelopes[1] $s_1(t)$ and $s_2(t)$ of the two simultaneously talking speakers through the Pearson correlation coefficient. For the sake of an easy exposition but without loss of generality, we here assume only two competing speakers, although all presented algorithms and procedures can be extended to more speakers.

In the remainder of the paper, we will adopt a matrix-vector notation, in which the decoder is written as $\mathbf{d} = \begin{bmatrix} d_1(0) & d_1(1) & \cdots & d_1(L-1) & d_2(0) & \cdots & d_C(L-1) \end{bmatrix}^{\text{T}} \in \mathbb{R}^{CL}$. Assume (for now) the availability of K training segments of T time samples, where the available training information is described as $\{\mathbf{X}_k, (\mathbf{s}_{1_k}, \mathbf{s}_{2_k}), y_k\}_{k=1}^{K}$, containing an EEG data $\mathbf{X}_k$ matrix (collecting all T time samples in training segment k; a rigorous definition is given in (3)), speech envelopes $\mathbf{s}_{1_k} \in \mathbb{R}^T$ and $\mathbf{s}_{2_k} \in \mathbb{R}^T$ (similarly), and attention labels $y_k \in \{1, 2\}$, indicating which speech envelope ($\mathbf{s}_{1_k}$ or $\mathbf{s}_{2_k}$) is the attended one ($\mathbf{s}_{\text{a}_k}$). For each training segment k, the attended speech envelope is determined as

$$\mathbf{s}_{\text{a}_k} = \begin{cases} \mathbf{s}_{1_k} & \text{if } y_k = 1, \\ \mathbf{s}_{2_k} & \text{if } y_k = 2. \end{cases} \tag{1}$$

[1]The speech envelope can be defined and extracted in different ways [9]. In this paper, we use the speech envelope extract procedure based on a gammatone filterbank and power-law compression as proposed in [9].

The decoder is then trained by minimizing the squared error between the actual attended and reconstructed speech envelope across all training segments:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmin}} \sum_{k=1}^{K} ||\mathbf{s}_{\text{a}_k} - \mathbf{X}_k \mathbf{d}||_2^2 = \underset{\mathbf{d}}{\operatorname{argmin}} ||\mathbf{s}_{\text{a}} - \mathbf{X}\mathbf{d}||_2^2, \tag{2}$$

with $\mathbf{s}_{\text{a}} = \begin{bmatrix} \mathbf{s}_{\text{a}_1}^{\text{T}} & \cdots & \mathbf{s}_{\text{a}_K}^{\text{T}} \end{bmatrix}^{\text{T}} \in \mathbb{R}^{KT}$ the concatenated actual attended speech envelope and where the block Hankel matrix $\mathbf{X} \in \mathbb{R}^{KT \times CL}$ represents the concatenated time-lagged C-channel EEG with L time lags:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_K \end{bmatrix}, \mathbf{X}_k = \begin{bmatrix} \mathbf{X}_{k,1} & \cdots & \mathbf{X}_{k,C} \end{bmatrix} \in \mathbb{R}^{T \times CL}, \tag{3}$$

with $\mathbf{X}_{k,c} \in \mathbb{R}^{T \times L}$ a Hankel matrix containing the time-lagged EEG data of the kth training segment and c^{th} EEG channel:

$$\mathbf{X}_{k,c} = \begin{bmatrix} x_{k,c}(0) & x_{k,c}(1) & \cdots & x_{k,c}(L-1) \\ x_{k,c}(1) & x_{k,c}(2) & \cdots & x_{k,c}(L) \\ \vdots & \vdots & & \vdots \\ x_{k,c}(T-1) & 0 & \cdots & 0 \end{bmatrix}.$$

$\mathbf{X}\hat{\mathbf{d}} = \hat{\mathbf{s}}_{\text{a}} \in \mathbb{R}^{KT}$ then represents the reconstructed speech envelope over all training segments. The solution of (2) is found by solving the normal equations:

$$\hat{\mathbf{d}} = \mathbf{R}_{xx}^{-1} \mathbf{r}_{xs}, \tag{4}$$

with

$$\mathbf{R}_{xx} = \mathbf{X}^{\text{T}}\mathbf{X} = \sum_{k=1}^{K} \mathbf{X}_k^{\text{T}} \mathbf{X}_k \in \mathbb{R}^{CL \times CL}$$

the estimated EEG autocorrelation matrix and

$$\mathbf{r}_{xs} = \mathbf{X}^{\text{T}} \mathbf{s}_{\text{a}} = \sum_{k=1}^{K} \mathbf{X}_k^{\text{T}} \mathbf{s}_{\text{a}_k} \in \mathbb{R}^{CL} \tag{5}$$

the estimated crosscorrelation vector between the EEG and the attended speech envelope. It is important to notice that only in (5) we need the attention labels y_k to select the attended speech envelope $\mathbf{s}_{\text{a}_k}$ in segment k (see (1)). We use shrinkage to regularize the estimated autocorrelation matrix:

$$\mathbf{R}_{xx} = (1-\lambda)\mathbf{X}^{\text{T}}\mathbf{X} + \lambda \frac{\text{Tr}\left(\mathbf{X}^{\text{T}}\mathbf{X}\right)}{CL} \mathbf{I}, \tag{6}$$

with $\mathbf{I} \in \mathbb{R}^{CL \times CL}$ the identity matrix and where the shrinkage parameter $0 \leq \lambda \leq 1$ is analytically determined [10], [11]:

$$\lambda = \min \left(\frac{\sum_{k=1}^{K} \sum_{t=1}^{T} \left|\left| \mathbf{x}_{k,t}\mathbf{x}_{k,t}^{\text{T}} - \mathbf{X}^{\text{T}}\mathbf{X} \right|\right|_{\text{F}}^2}{\text{Tr}\left(\left(\mathbf{X}^{\text{T}}\mathbf{X}\right)^2 \right) - \frac{\text{Tr}(\mathbf{X}^{\text{T}}\mathbf{X})^2}{CL}}, 1 \right),$$

with $\mathbf{x}_{k,t}^{\text{T}} \in \mathbb{R}^{CL}$ the tth row of the matrix $\mathbf{X}_k$.

Given the estimated decoder $\hat{\mathbf{d}}$ and T_{test} time samples of a new EEG segment $\mathbf{X}^{(\text{test})} \in \mathbb{R}^{T_{\text{test}} \times CL}$ of a subject listening to one out of two competing speakers with speech envelopes $\mathbf{s}_1^{(\text{test})}$ and $\mathbf{s}_2^{(\text{test})}$, a decision about the auditory attention of the listener can be made by:

1) reconstructing the attended speech envelope from the EEG: $\hat{\mathbf{s}}^{(\text{test})} = \mathbf{X}^{(\text{test})}\hat{\mathbf{d}}$ and
2) computing the Pearson correlation coefficients $\rho\left(\hat{\mathbf{s}}^{(\text{test})}, \mathbf{s}_1^{(\text{test})}\right)$ and $\rho\left(\hat{\mathbf{s}}^{(\text{test})}, \mathbf{s}_2^{(\text{test})}\right)$ between this reconstructed speech envelope and the original speech envelopes. The speaker corresponding to the highest correlation coefficient is identified as the attended speaker.

For stimulus reconstruction, there is an important tradeoff between the accuracy of the decision and the decision segment length T_{test}, i.e., the number of time samples used to make a decision [5], [12]. A longer decision segment leads to more accurate estimates of the Pearson correlation coefficients, thereby improving accuracy on the AAD decisions. However, this comes with the drawback of a poorer time resolution at which the AAD decisions are made, due to the longer decision segment length.

B. Unsupervised stimulus reconstruction

Let us now assume that the attention labels $\{y_k\}_{k=1}^K$ are not known, even during training, meaning that we do not know to which of the speakers the subject is attending when designing our decoder (i.e., using (1) has become impossible). If we indeed can train the decoder without these labels, this would avoid the need for a dedicated training session during which the subject is instructed to attend to a specific speaker in order to collect ground-truth data. As a result, performing AAD becomes an *unsupervised* classification problem. The absence of labels is a roadblock in the computation of the crosscorrelation vector in (5), which requires the use of the correct speech envelope in its calculation (using the unattended envelope in (5) would result in a decoder that emphasizes the wrong speaker).

In [8], we proposed an unsupervised batch-training procedure on an unlabeled dataset $\{\mathbf{X}_k, (\mathbf{s}_{1_k}, \mathbf{s}_{2_k})\}_{k=1}^K$ of K segments (i.e., attention labels $\{y_k\}_{k=1}^K$ are unavailable/unknown). The main idea is to iteratively retrain a decoder by using labels that are predicted by the decoder from the previous iteration. A short description of this iterative procedure is as follows. We start with an initial decoder $\hat{\mathbf{d}}^{(0)}$ at iteration 0, which can be, e.g., a pre-computed subject-independent decoder or even a decoder with random entries. First, the EEG autocorrelation matrix $\mathbf{R}_{xx}$ is estimated as in (6), which does not require any attention labels. The iterative prediction of the labels and updating of the decoder then comprises the following steps:

1) Given a decoder $\hat{\mathbf{d}}^{(i)}$ that is available after the ith iteration, apply it on each EEG segment $\mathbf{X}_k$ to reconstruct the attended speech envelope:
$$\forall k : \hat{\mathbf{s}}_{\text{a}_k}^{(i)} = \mathbf{X}_k \hat{\mathbf{d}}^{(i)}.$$
2) Per segment, correlate the reconstructed speech envelope $\hat{\mathbf{s}}_{\text{a}_k}^{(i)}$ with both speech envelopes $(\mathbf{s}_{1_k}, \mathbf{s}_{2_k})$ to predict the attended speaker. As before, the first speaker is identified as the attended one if the sample Pearson correlation coefficient $\rho\left(\hat{\mathbf{s}}_{\text{a}_k}^{(i)}, \mathbf{s}_{1_k}\right) > \rho\left(\hat{\mathbf{s}}_{\text{a}_k}^{(i)}, \mathbf{s}_{2_k}\right)$ and vice versa. The predicted attended envelope is denoted as $\mathbf{s}_{\text{pred}_k}^{(i)} \in \{\mathbf{s}_{1_k}, \mathbf{s}_{2_k}\}$.
3) Using the EEG segments and corresponding speech envelopes of the predicted attended speaker $\{\mathbf{s}_{\text{pred}_k^{(i)}}\}_{k=1}^K$, the crosscorrelation vector can be computed/updated as in (5). It is crucial to use the *original* speech envelope ($\mathbf{s}_{1_k}$ or $\mathbf{s}_{2_k}$) and not the envelope $\hat{\mathbf{s}}_{\text{a}_k}^{(i)}$ that was reconstructed from the EEG. Given the new crosscorrelation vector, the decoder $\hat{\mathbf{d}}^{(i+1)}$ can be updated as in (4). Then return to step (i) and iterate until convergence.

This iterative unsupervised predicting of labels could potentially inject incorrect labels and thus incorrect data in the estimation of the decoder, which could in principle lead to a downward spiral of incorrect updating and thus to a badly-performing decoder. Remarkably, it was shown in [8] that a *self-leveraging effect* occurs in this batch-mode iterative updating where the new decoder outperforms the previous decoder, despite the presence of labeling errors, resulting in an upward spiral instead of a downward spiral. This happens even in the case where the initial decoder is initialized with random values. This unsupervised subject-specific decoder outperformed a supervised subject-independent decoder (i.e., trained on data from other subjects than the one under test) and even closely approximated the performance of a supervised subject-specific decoder [8].

III. Time-adaptive unsupervised stimulus reconstruction for AAD

The unsupervised training procedure of Section II-B assumes the availability of multiple data segments at once (i.e., batch-training). The batch computation is inherent to the procedure; once a new decoder is computed, *all* labels in the recording are repredicted to improve the next decoder. After the unsupervised batch training, the final decoder is fixed and applied to unseen data from the subject under test. However, such a pre-trained time-invariant decoder does not adapt to non-stationarities due to changing conditions and situations and may thus perform suboptimally. Here, we propose a time-adaptive realization of such an unsupervised AAD decoder, i.e., a decoder that adapts itself over time when EEG and audio data (processed in envelopes) are continuously streaming in.

Assume some initial decoder. Data segments of T_{ud} samples of EEG and audio data start streaming in. At a certain point in time, assume the kth segment of EEG data $\mathbf{X}_k \in \mathbb{R}^{T_{\text{ud}} \times CL}$ (see (3)) and corresponding segments of the speech envelopes of the two competing speakers $(\mathbf{s}_{1_k}, \mathbf{s}_{2_k}) \in \mathbb{R}^{T_{\text{ud}}}$ become available. There is no information available about which speaker is the attended or unattended one in this segment. The goal is now to update the decoder in an unsupervised manner based on the newly available information, to which end we will propose and compare two approaches (Section III-A and III-B).

In this time-adaptive procedure, it is important to distinguish the updating segment length T_{ud} from the decision segment length T_{test}. The former, equivalent to the segment length T in the previous sections, corresponds to the length of the segments on which the prediction of the labels for the

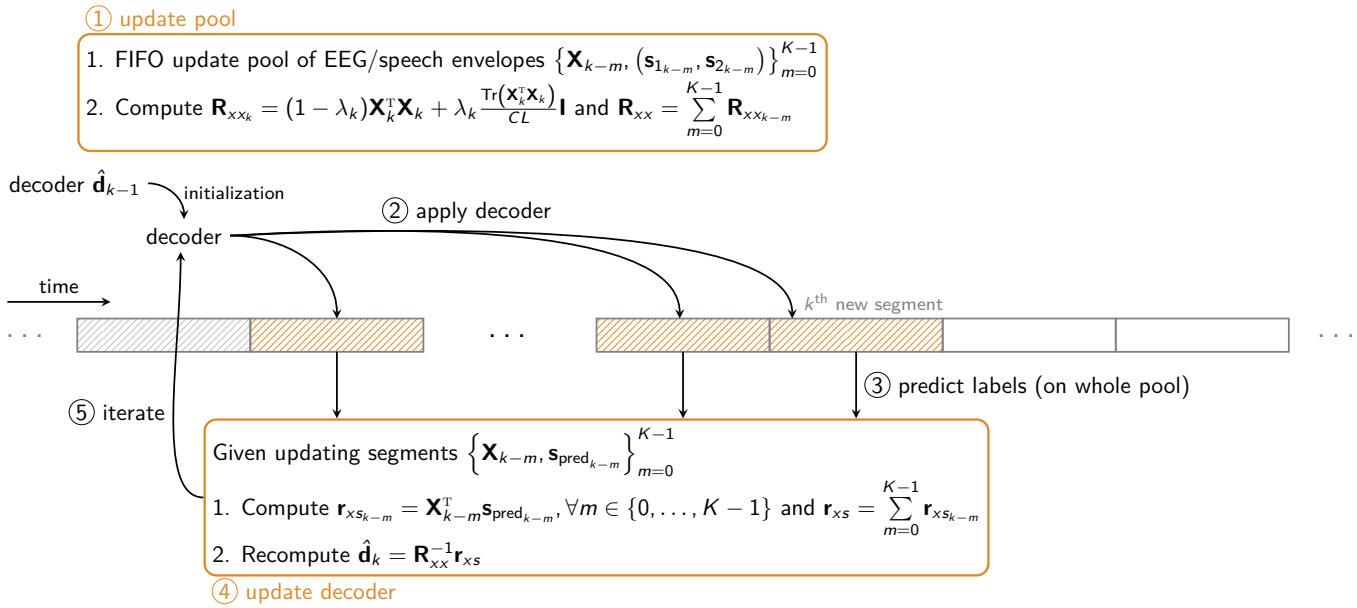

Figure 1: The time-adaptive unsupervised sliding window scheme to update a stimulus decoder, with repredictions of the labels on previous segments.

updating/training and the updating/training itself is performed. The latter corresponds to the length of the segments on which AAD decisions are made to in the end steer the enhancement algorithm in the hearing device. This decision segment length is, therefore, much more sensitive to speed (e.g., because of switches in auditory attention [12]) than the updating segment length, as there can be some delay allowed in updating the decoder. Therefore, the updating segment length is typically larger than the decision segment length $T_{\text{ud}} \geq T_{\text{test}}$, i.e., within each updating segment, multiple AAD decisions are made.

In the time-adaptive sliding window approach (Section III-A), the aforementioned batch-mode procedure is mimicked (i.e., including repredictions of the labels of previous segments) but over a finite time horizon which is implemented as a sliding window. In Section III-B, we propose an alternative time-adaptive approach that does *not* recompute previously predicted labels and, therefore, can be implemented in a recursive manner. This is much more attractive from a computational and memory usage point of view. Both approaches to update the decoder are explained in more detail in the following sections.

A. Sliding window implementation

In the sliding window implementation (Figure 1), a pool of K data segments of updating segment length T_{ud} is permanently kept in memory using a first in, first out (FIFO) principle. When a new data segment $\{\mathbf{X}_k, (\mathbf{s}_{1_k}, \mathbf{s}_{2_k})\}$ becomes available, the oldest data segment is discarded and the pool is updated with the newest one, resulting in the new pool $\left\{\mathbf{X}_{k-m}, \left(\mathbf{s}_{1_{k-m}}, \mathbf{s}_{2_{k-m}}\right)\right\}_{m=0}^{K-1}$. The stimulus decoder is then updated similarly to the batch-mode implementation explained in [8] and Section II-B, but on the finite pool of K segments.

First, the EEG autocorrelation matrix of the new segment is computed similarly to (6):

$$\mathbf{R}_{xx_k} = (1-\lambda_k)\mathbf{X}_k^{\mathrm{T}}\mathbf{X}_k + \lambda_k \frac{\mathrm{Tr}\left(\mathbf{X}_k^{\mathrm{T}}\mathbf{X}_k\right)}{CL}\mathbf{I}, \tag{7}$$

with the regularization parameter λ_k recomputed per new segment k, such that the aggregated autocorrelation matrix across the whole pool of K segments can be updated/recomputed:

$$\mathbf{R}_{xx} = \sum_{m=0}^{K-1} \mathbf{R}_{xx_{k-m}}. \tag{8}$$

The autocorrelation matrices of the previous segments can be recomputed or stored and retrieved from previous computations. We found empirically that better results are obtained when regularizing the new autocorrelation matrix (7) *before* being stored and combined in (8), instead of regularizing the combined autocorrelation matrix (8), i.e., *after* combining the autocorrelation matrices from the different segments.

Using the decoder $\hat{\mathbf{d}}_{k-1}$ from the previous step, the iterative procedure of predicting labels, updating the crosscorrelation vector(s) and decoder on the pool of K segments $\left\{\mathbf{X}_{k-m}, \left(\mathbf{s}_{1_{k-m}}, \mathbf{s}_{2_{k-m}}\right)\right\}_{m=0}^{K-1}$ can be initiated. Given the per-segment predicted attended speaker $\{\mathbf{s}_{\text{pred}_{k-m}}\}_{m=0}^{K-1}$ (initially obtained using $\hat{\mathbf{d}}_{k-1}$), the crosscorrelation vectors can be updated as in (5):

$$\mathbf{r}_{xs_{k-m}} = \mathbf{X}_{k-m}^{\mathrm{T}}\mathbf{s}_{\text{pred}_{k-m}}, \forall m \in \{0, \ldots, K-1\}.$$

The aggregated crosscorrelation vector and corresponding decoder can then be computed as:

$$\mathbf{r}_{xs} = \sum_{m=0}^{K-1} \mathbf{r}_{xs_{k-m}} \Rightarrow \hat{\mathbf{d}}_k = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xs}.$$

This predict-and-update procedure is then iterated I times on the same pool of K segments. We choose $I = 5$, which showed to be sufficient for convergence in [8]. Given that

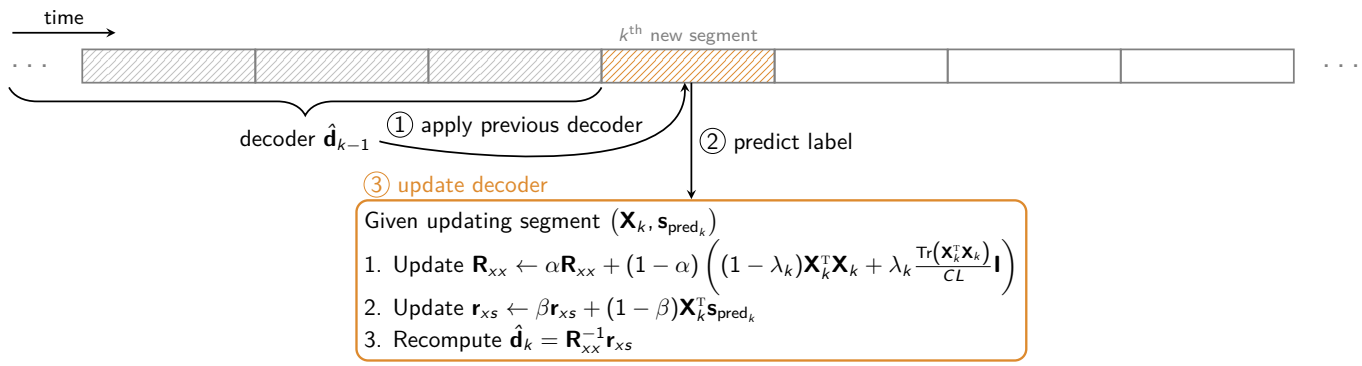

Figure 2: The time-adaptive unsupervised recursive predict-and-update scheme to update a stimulus decoder.

we iterate over this pool of K segments, this approach can only be implemented in a sliding window manner and not recursively. Lastly, the pool size parameter K represents an important tradeoff between accuracy, adaptivity, and computational complexity and memory usage. A longer sliding window (i.e., larger K) means that more data is available to compute the decoder, resulting in a better approximation of the batch-mode decoder but also resulting in a lower adaptivity and higher memory requirements (see also Section IV).

B. Recursive implementation

As an alternative to the sliding window implementation, we propose a single-shot predict-and-update scheme (Figure 2). As opposed to the sliding window approach, the labels of previous updating segments are not repredicted, enabling a recursive implementation that is much more efficient from a computational and memory usage point of view. In this recursive implementation, the decoder resulting from the previous update $\hat{\mathbf{d}}_{k-1}$ is applied to the new segment $\{\mathbf{X}_k, (\mathbf{s}_{1_k}, \mathbf{s}_{2_k})\}$ of length T_{ud} to predict the label of the new segment, resulting in the predicted attended envelope $\mathbf{s}_{\text{pred}_k}$. To update the decoder, a regularized autocorrelation matrix is computed based on the new k^{th} segment as in (7), while the predicted attended envelope is used to compute a crosscorrelation vector as in (5):

$$\mathbf{r}_{xs_k} = \mathbf{X}_k^{\text{T}}\mathbf{s}_{\text{pred}_k}.$$

This new autocorrelation matrix $\mathbf{R}_{xx_k}$ and crosscorrelation vector $\mathbf{r}_{xs_k}$ can then be combined with the autocorrelation matrix $\mathbf{R}_{xx}$ and crosscorrelation vector $\mathbf{r}_{xs}$ integrating all previous information to update the decoder:

$$\begin{cases} \mathbf{R}_{xx} \leftarrow \alpha \mathbf{R}_{xx} + (1-\alpha)\mathbf{R}_{xx_k} \\ \mathbf{r}_{xs} \leftarrow \beta \mathbf{r}_{xs} + (1-\beta)\mathbf{r}_{xs_k} \end{cases} \Rightarrow \hat{\mathbf{d}}_k = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xs}.$$

The influence of the weighting parameters α and β will be empirically evaluated in Section IV.

Unlike the sliding window implementation, which uniformly weighs the K past segments in the new decoder, this recursive algorithm implements an exponential weighting across all past segments. This exponential weighting could be advantageous, especially in an adaptive context, as the more relevant closest (past) segments have higher weights than those further in the past. One can choose the weighting parameters α and β such that the center of mass of the exponential weighting is the same as of the sliding window approach [13]:

$$\alpha = \frac{K-1}{K+1} \text{ and } K = \frac{1+\alpha}{1-\alpha}. \tag{9}$$

A possible drawback of this recursive implementation is that previous labels are not repredicted, and that one can not apply multiple iterations over a pool of segments as in the sliding window version. This could lead to slower convergence or poorer accuracies. However, the upside is that the procedure is much easier to implement and much more efficient in terms of memory and computation resources (see Section III-C). In Section IV, we will demonstrate that the impact on convergence speed and accuracy is negligible.

C. Memory usage

1) Sliding window implementation: For the sliding window implementation, at least the pool of K EEG segments ($K \times T_{\text{ud}} \times C$) and $2K$ speech envelopes ($2K \times T_{\text{ud}}$) need to be permanently stored in memory. One does not need to store the L different time lags, as these can always be generated from the original EEG data. Furthermore, it is not possible to just store the autocorrelation matrices and crosscorrelation vectors of the previous EEG segments, which would require less memory usage, as for the repredictions of the labels, the decoder has to be applied to the original EEG and speech envelope data. This leads to the following memory usage:

$$(C+2)KT_{\text{ud}} \sim \mathcal{O}(KT_{\text{ud}}C),$$

where generally $T_{\text{ud}} >> K$ or C. As the storage of the sample dimension T_{ud} is required, this is generally a very high memory usage.

2) Recursive implementation: The recursive implementation minimally requires the permanent storage of one autocorrelation matrix $\mathbf{R}_{xx}$ (built from $\frac{CL(CL+1)}{2}$ elements due to symmetry) and one crosscorrelation vector $\mathbf{r}_{xs}$ (built from CL elements). This leads to the following memory usage:

$$CL + \frac{CL(CL+1)}{2} \sim \mathcal{O}(C^2L^2),$$

which is, as expected, much less than the sliding window approach.

To better appreciate the differences, consider the following practical realistic example with $C = 24$ EEG channels, $L = 6$

time lags, pool size $K = 19$, and an updating segment length T_{ud} equivalent to 1200 samples (corresponding to 60 s when the EEG and speech envelopes are downsampled to 20 Hz). The sliding window approach then requires the storage of 592 800 elements, which is more than 50 times more than for the recursive implementation, which requires to only story 10 584 elements.

IV. Validation and comparison

We test both versions of the time-adaptive unsupervised stimulus reconstruction algorithms of Section III for different hyperparameter settings (i.e., the updating segment length T_{ud}, pool size K in the sliding window implementation, and exponential forgetting factors α, β in the recursive implementation). In all experiments, we start from a different *fully random initial* decoder, generating the first prediction(s). In the recursive implementation, the initial autocorrelation matrix and crosscorrelation vector are initialized with all zeros. We compare the sliding window and recursive implementation and select a set of hyperparameters on one dataset and validate the chosen algorithm on a second one. These datasets are introduced in Section IV-A, while the performance metrics are described in Section IV-B. The experiments and results are discussed in Section IV-C, with a more detailed discussion on the effect of repredictions of the labels in the sliding window approach in Section IV-D. The final settings are validated on the second dataset in Section IV-E.

A. Data and preprocessing

1) AAD datasets: The first AAD dataset (Dataset I) is from [9] and contains the EEG (64-channel BioSemi ActiveTwo system) and audio data of 16 normal-hearing subjects participating in an AAD experiment, where the subjects were instructed to listen to one of two competing speakers. Per subject, 8 stories of 6 min and 12 repetitions of 2 min of those stories are presented, resulting in 72 min data per subject. This dataset is available online [14]. The second AAD dataset (Dataset II) is from [15] and will act as an independent validation dataset. It contains the EEG (64-channel BioSemi ActiveTwo system) and audio data of 18 normal-hearing subjects in a similar AAD experiment with two competing speakers, using different acoustic room properties. Per subject, 50 min of data (60×50 s trials) are available. This dataset is also available online [16]. For more details on both datasets, we refer to the corresponding papers [9] and [15].

2) Preprocessing: To preprocess the EEG and audio data, we applied the same preprocessing steps as in [8], [9]. The speech signals are first filtered using a gammatone filterbank. Using a power-law operation with exponent 0.6, an envelope is computed for each subband signal. All subband envelopes are afterwards summed to one envelope. Both EEG and speech envelopes are filtered between $1 - 9$ Hz and downsampled to 20 Hz (Dataset I)/32 Hz (Dataset II).

B. Performance metrics

To evaluate a specific implementation with a specific set of hyperparameters, we use three performance metrics, quantifying the accuracy, adaptivity, and memory usage of each algorithm:

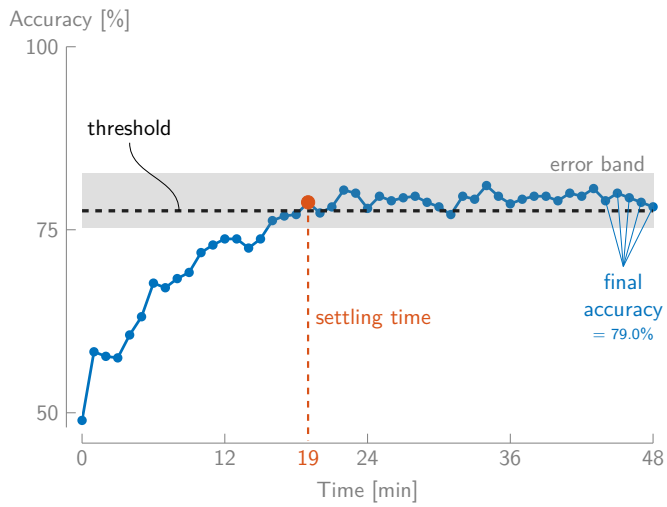

Figure 3: Illustration of the adaptation curve and performance metrics for a representative subject (Subject 4) of Dataset I, starting from a random initial decoder and updating every 60 s, for the recursive implementation with $\alpha = \beta = 0.9$.

- *Final accuracy:* the final accuracy is defined as the average of the accuracies on the test set across the last 5 min of updating, i.e., after the adaptive decoder has had sufficient time to converge to a steady-state regime.
- *Settling time:* in a time-adaptive context, not only accuracy but also adaptivity or speed of adaptation is an important metric. Here, we quantify the adaptivity with the settling time, defined similarly as in control theory [17]. This settling time is defined as the point in time where the accuracy has reached a threshold for the first time *and* remains within a predefined error band for the remainder of the updating procedure. The threshold is defined as a convex combination of the final accuracy and the initial chance level performance (50%; before the updating procedure):

$$\text{threshold} = 0.95 \times \text{final accuracy} + 0.05 \times 0.5.$$

  The error band, which allows taking the variability into account, is defined as:

$$\text{error band} = \text{final accuracy} \pm 2M,$$

  with M the difference between the maximum and minimum across the last 5 min accuracies.
- *Memory usage:* the memory usage, i.e., the number of elements that need to be permanently stored in memory, is computed as in Section III-C.

Figure 3 illustrates the final accuracy and settling time performance metrics for a representative subject and specific implementation (this figure is only meant to illustrate the updating procedure and the definitions of the performance metrics, and should not be viewed as a validation result).

C. Hyperparameter selection

We test the different implementations of Section III for different hyperparameter settings on Dataset I. Per subject, we randomly permute the 6 min trials of the first 48 min and use

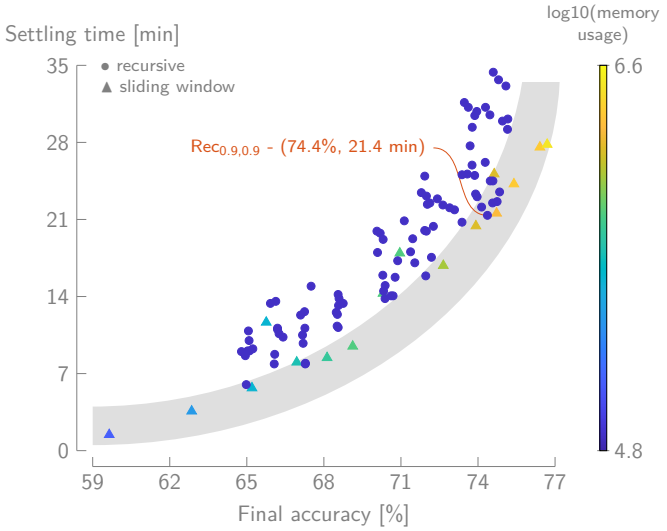

Figure 4: Settling time vs. final accuracy for different parameter settings. The shaded area highlights the points that are close to or in the Pareto front. The indicated recursive algorithm with $\alpha = \beta = 0.9$, starting from a random initial decoder, gives one of the best tradeoffs between final accuracy, adaptivity, and memory usage.

those as the updating set, i.e., the data on which the time-adaptive unsupervised updating from a random initial decoder is performed. To track the accuracy of the updated decoder over time, after each update, we evaluate the decoder on the separate set of the last $24\,\text{min}$ of repetition data, using $T_{\text{test}} = 30\,\text{s}$ decision segments to make a decision about the auditory attention (i.e., to compute the Pearson correlation coefficient with both speech envelopes). Per subject, we perform 10 random permutations. For the decoder, we choose time lags up to $250\,\text{ms}$ [6], [9], which corresponds to $L = 6$ for Dataset I and $L = 9$ for Dataset II (as both are sampled at different rates). We choose updating segment length $T_{\text{ud}} = 60\,\text{s}$ (different from decision segment length $T_{\text{test}} = 30\,\text{s}$), i.e., we update the decoder every $60\,\text{s}$. As the performance of the stimulus decoder heavily depends on the amount of data available to make a decision [5], [12], similarly to [8], we choose T_{ud} as large as possible - without waiting too long - to produce as reliable labels as possible. As explained in the introduction of Section III, we can afford such a longer delay in updating (as opposed to the decision segment length, explaining why we work on different time resolutions for the updating and testing). However, we do not take T_{ud} larger than $60\,\text{s}$, as the performance of the stimulus decoder starts to saturate above this segment length, and because this would require a too long sustained attention on the same speaker.

Figure 4 shows the average final accuracy, settling time, and memory usage for different settings of the sliding window and recursive implementations across the 16 subjects of Dataset I. The sliding window implementation is evaluated for different updating segment lengths ($T_{\text{ud}} \in \{60, 30, 10\,\text{s}\}$) and pool sizes ($K \in \{10, 20, \ldots, 60\}$, except for $T_{\text{ud}} = 60\,\text{s}$, where the maximum is $K = 40$). Only the results for $T_{\text{ud}} = 60\,\text{s}$ updating segments are shown, which are found to be superior to $T_{\text{ud}} = 30$ and $10\,\text{s}$. Therefore, and for the clarity of Figure 4, the recursive implementation is only evaluated for $T_{\text{ud}} = 60\,\text{s}$ and α, β ranging independently from each other from 0.6 to 0.95 in steps of 0.05 and $\alpha = \beta$ ranging in more fine-grained steps of 0.005 from 0.8 to 0.95. As can be seen in Figure 4, the memory usage of the recursive implementation is the same for every setting (see Section III-C2), while this is dependent on the pool size K and updating segment length T_{ud} for the sliding window implementation (see Section III-C1). In general, there is a clear positive correlation between a higher final accuracy and a higher settling time, representing the tradeoff between accuracy and adaptability of the decoder. The points in the shaded lower envelope area of the point cloud in Figure 4 represent the Pareto front, as they give the best tradeoffs between high final accuracy and low settling time.

Surprisingly, the Pareto front of the recursive implementations (i.e., without repredictions of previous labels) seems to achieve very similar performances in terms of final accuracy and settling time as the sliding window implementations (i.e., with repredictions of previous labels). In Section IV-D, we will investigate more closely why these repredictions of previous labels seem to have such little effect. Moreover, the recursive implementation requires on average $16\times$ less memory than the sliding window implementation (see Figure 4) and is computationally more efficient, making it the preferred implementation.

As indicated in Figure 4, a good choice is the recursive implementation with $\alpha = \beta = 0.9$, resulting on average in 74.4% final accuracy (standard deviation 12.1%) after $21.4\,\text{min}$ (st. dev. $11.7\,\text{min}$). The latter implies that it takes about $20\,\text{min}$ before the decoder has learned how to optimally decode the attended speaker. These settings are therefore used in the remainder of the experiments. Although there are a few settings of the sliding window implementation (i.e., $T_{\text{ud}} = 60\,\text{s}$ and $K = 40$) that give a slightly better final accuracy for similar settling times, they require more memory storage ($42\times$ more elements) and are also computationally heavier (due to the repredictions of the labels). Lastly, using the formula in (9), $\alpha = \beta = 0.9$ corresponds to $K = 19\,\text{min}$. This is indeed consistent with the results of the sliding window implementation with $K = 19$, which has a very similar performance (73.6% (st. dev. 13.2%) in $19.4\,\text{min}$ (st. dev. $11.5\,\text{min}$)), while requiring much more memory. There is no noticeable benefit from the exponential weighting over the uniform weighting, given that the performance is tested on an asynchronous, independent test set. In Section V, we will concurrently test and update on the same data, potentially revealing the benefit of exponential weighting.

D. Effect of repredictions

The results in Section IV-C show that the single-shot recursive implementation, without repredictions of the labels, performs on par with the sliding window implementation with repredictions of the labels of previous segments. This suggests that, in the considered time-adaptive context, the iterative repredictions of labels on the current pool of K segments have no added benefit and that the labels hardly change between before and after the relabeling procedure. This is confirmed by computing the total number of labels that changed between before and after the iterative relabeling

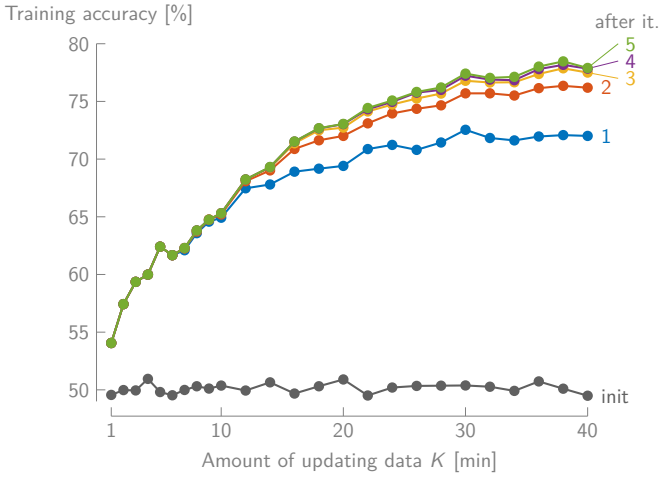

Figure 5: The training accuracy on $T_{\text{test}} = 30\,\text{s}$ decision segments for the batch-mode unsupervised iterative updating procedure as a function of the amount of updating data, for different numbers of relabeling iterations ('init' refers to the accuracy when no iterations are performed). Given that the updating segment length $T_{\text{ud}} = 60\,\text{s}$, the amount of updating data corresponds to pool size K (in number of minutes).

procedure in all updates before the settling time, i.e., before reaching steady-state performance. For $T_{\text{ud}} = 60\,\text{s}$, this total number of labels that changed is (on average across subjects and random permutations in Section IV) only 0.16 for $K = 10$, 0.70 for $K = 20$, and 1.13 for $K = 40$. This shows that the number of self-corrected labels in the iterative relabeling procedure is minimal, even more so if the pool size K is small.

To more closely investigate this dependence of the relabeling on the pool size K, we compute the training accuracy (i.e., the percentage of correct labels of the updating set) in the different iterations as a function of the size of the updating set for the batch-mode unsupervised algorithm in [8] using $T_{\text{ud}} = 60\,\text{s}$ updating segments and $T_{\text{test}} = 30\,\text{s}$ decision segments. Per subject and size of the updating set, 10 runs with different randomly selected updating sets of size K from the total dataset are performed. Figure 5 then shows the average training accuracy across subjects and runs on Dataset I. From Figure 5, it is clear that the self-correcting behavior on the predicted labels of the first iteration only starts to occur when the updating set contains more than $14 - 20\,\text{min}$ of data. This can be explained from a mathematical point of view as some sort of overfitting effect: the decoder has enough degrees of freedom to span all initial predictions in the subsequent iterations when K is small, leading to an overfitted decoder.

When $K \geq 14$, there is a clear effect of the second and subsequent iterations (until convergence to the fixed point [8]). This effect, however, seems not to be present in the time-adaptive context. This is explained by considering the initial decoder for each new decoder update when a new segment becomes available. In the batch-mode design, this initial decoder is always a random decoder, whereas in the time-adaptive context, this will only be the case for the first received data segment. In the later updates, the initial decoder is already improved based on past data. Consider the case of $T_{\text{ud}} = 60\,\text{s}, K = 20$ in Figure 5. In the time-adaptive case, already 19 updates will have been completed before a full pool of $K = 20$ segments becomes available. After 19 updates, however, the decoder has already substantially improved (see also Figure 3). Therefore, as suggested in Figure 5, the decoder will not exhibit random performance but performance close to the one of the converged decoder in Figure 5. Consequently, the effect of a reprediction on the labels on previous segments in the pool will be similar to the effect of the last iterations (3, 4, 5) in Figure 5, that is, very small. In other words, the initial decoder is then already close to the fixed point of the updating procedure [8].

Given the important tradeoff in memory usage and computational complexity, these insignificant improvements do not outweigh the additional required resources.

E. Validation on an independent dataset

To confirm that the recursive implementation with $T_{\text{ud}} = 60\,\text{s}$ and $\alpha = \beta = 0.9$ is a robust choice across subjects and datasets and that no overfitting of the hyperparameters has occurred on Dataset I, we apply the recursive algorithm on the completely independent Dataset II, again starting from a fully random decoder. Given that Dataset II only contains $50\,\text{s}$ trials, we choose $T_{\text{ud}} = 50\,\text{s}$. Furthermore, given that $\alpha = \beta = 0.9$ is equivalent with $K = 19$ according to (9), resulting in a $19\,\text{min}$ history when using $60\,\text{s}$ segments, this becomes $K = 22.8$ for $50\,\text{s}$ segments. Using (9), this means that the equivalent choice here for Dataset II is $\alpha = \beta = 0.916$.

We test this recursive implementation with $T_{\text{ud}} = 50\,\text{s}$ and $\alpha = \beta = 0.916$ on each of the 18 subjects of Dataset II by 10 times randomly selecting $40\,\text{min}$ as the updating set and the remaining $10\,\text{min}$ as test set. The average final accuracy and settling time (on $30\,\text{s}$ decision segments) are 76.2% (st. dev. 12.4%) and $18.8\,\text{min}$ (st. dev. $10.3\,\text{min}$). As this is very similar to the performance obtained in Section IV-C (and even slightly better), it confirms that the chosen specific recursive implementation is a robust choice.

V. Proof of concept

In Section IV, we tested the proposed time-adaptive unsupervised stimulus reconstruction algorithm asynchronously, i.e., the test set is time-independent from the updating set. Furthermore, the EEG data per subject are recorded in one session, with only small breaks in between. While these experiments allowed to investigate the behavior of the proposed method, they do not necessarily reflect a practical use case of the algorithm. In this section, we evaluate the proposed method in a realistic scenario, where the decoder is applied as in a practical neuro-steered hearing device application, i.e., while needing to adapt across multiple days of recordings and simultaneously providing AAD decisions.

A. Data and preprocessing

1) AAD dataset: We use a third dataset containing EEG and audio data of two subjects from a longitudinal AAD experiment across multiple days, carried out at the participants' homes. A two-talker AAD experiment was conducted in 8 different sessions that took place on 7 different days. In each

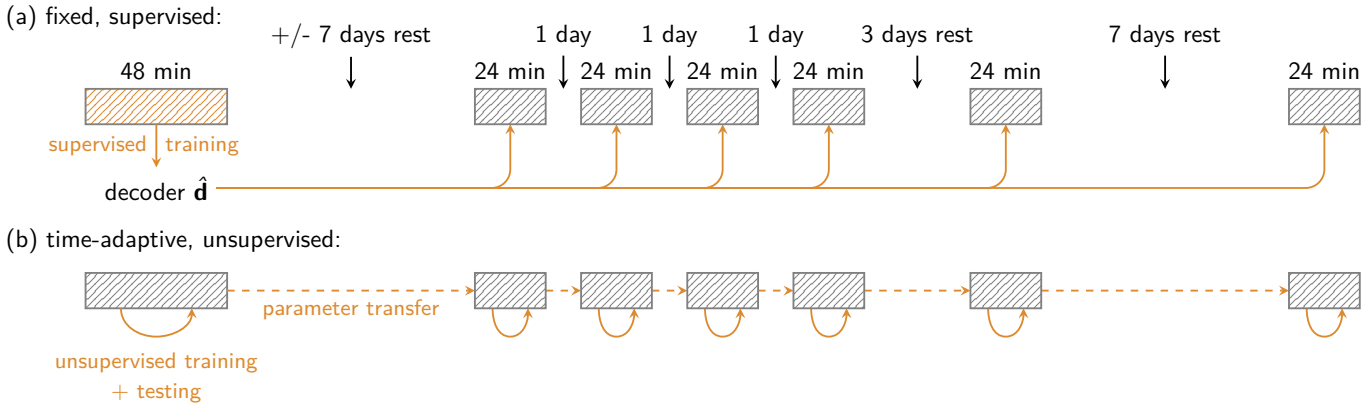

Figure 6: (a) The fixed decoder is trained in a supervised manner on the first two sessions on the first day and tested on all other sessions on the other days. (b) The proposed time-adaptive unsupervised decoder is initialized with a random decoder and is continuously updated over time and tested/applied on/to each next decision segment. The autocorrelation matrix and crosscorrelation vector are stored in between sessions ('parameter transfer').

session, 4 blocks of $6\,\text{min}$ stories are presented to each subject, resulting in a total of $192\,\text{min}$ of AAD data per subject. The audio stimuli differed across all sessions. The first two sessions took place on the same day, while the other 6 sessions took place on different days (see also Figure 6). The EEG was measured using a 24-channel SMARTING mobile EEG system from mBrainTrain. More detailed information about this dataset can be found in [18].

While this dataset was initially recorded for the purposes of neurofeedback experiments, it can be used to test the proposed time-adaptive unsupervised decoder as it reflects the practical use case of a neuro-steered hearing device. The algorithm will need to adapt over various days, meaning that there will be changes, for example, in EEG setup, electrode impedances, conditions, speaker and story characteristics, and state of mind of the user, as would all occur in practice. While two subjects are not enough to draw firm (statistical) conclusions as in the experiments in Section IV, it allows showcasing how the proposed algorithm can be used in a practical, online context.

2) Preprocessing: The EEG data and audio envelopes are preprocessed in the same way as in Section IV-A. The only difference is that $L = 400\,\text{ms}$ is chosen as the post-stimulus range of time lags for the decoder, reflecting the choice in [18].

B. Experiment

We compare the state-of-the-art supervised but fixed decoder with the proposed unsupervised time-adaptive decoder. In this section, we only consider the recursive version of the time-adaptive decoder, as it performs similarly to the sliding window version, while requiring much less resources. As shown in Figure 6, the supervised, fixed decoder is trained on the $48\,\text{min}$ of data from the first two sessions on the first day, using the information about which speaker is the attended one. This fixed decoder is then applied per $T_{\text{test}} = 30\,\text{s}$ decision segment on all other sessions on the other days. As such, this decoder reflects the practical use case where first data of a new neuro-steered hearing device user need to be recorded in an a priori calibration session, whereafter the trained decoder is loaded onto the device.

The time-adaptive unsupervised decoder is implemented using the settings as determined in Section IV and is initialized with a fully random decoder, while the autocorrelation matrix and crosscorrelation vector are initialized with all zeros. Per $T_{\text{ud}} = 60\,\text{s}$, the decoder is continuously updated using the recursive implementation with $\alpha = \beta = 0.9$. To fully leverage the time-adaptivity of this decoder, after each update, it is applied to the next two $T_{\text{test}} = 30\,\text{s}$ decision segments to make AAD decisions[2] (Figure 6). The first $48\,\text{min}$ of the first two sessions on the first day are used to let the decoder initialize and converge as in Figure 3, starting from a random initial decoder. In between sessions, as one would do in a practical scenario as well, the current autocorrelation matrix $\mathbf{R}_{xx}$ and crosscorrelation vector $\mathbf{r}_{xs}$ are re-used (hence the 'parameter transfer' in Figure 6) and not re-initialized each time from scratch.

Per $30\,\text{s}$ decision segment, a decision about the attended speaker is made, resulting in a binary correct/incorrect decision. To provide an comprehensible plot when plotting the accuracy over time, these binary decisions are smoothed using a 29-point moving average (i.e., per segment taking the past and following $7\,\text{min}$ into account). To provide a fair comparison between the fixed supervised and time-adaptive unsupervised decoder, the total accuracy is computed as the average over all binary decisions across all but the first $48\,\text{min}$ of the first two sessions on the first day.

C. Results

Figure 7 shows the smoothed accuracy as a function of time of both the fixed supervised and time-adaptive unsupervised algorithm for both test subjects. As explained before, there is no test accuracy present in the first $48\,\text{min}$ for the fixed supervised decoder, as it is trained on those sessions. During those first two sessions, the time-adaptive unsupervised decoder converges after $\pm 25\,\text{min}$, starting from a random initial decoder. This is more or less in line with the results of Section IV-C and IV-E.

[2]Two $30\,\text{s}$ decision segments as the decoder is only being updated every $60\,\text{s}$.

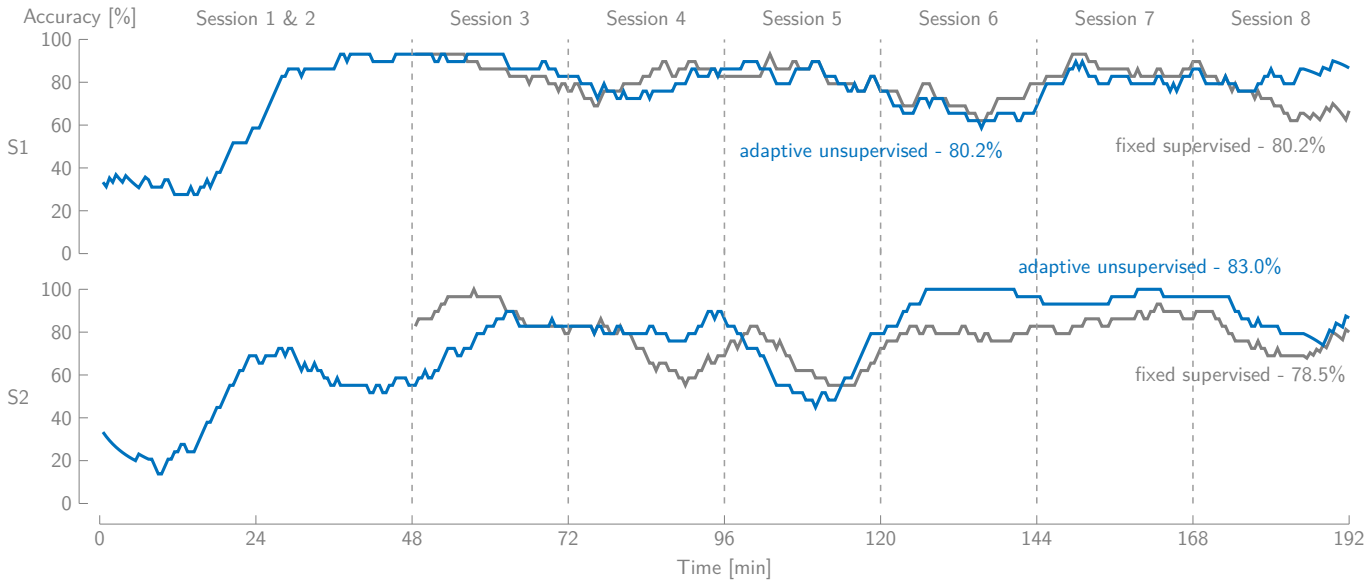

Figure 7: The smoothed accuracy as a function of time of the fixed supervised and adaptive unsupervised decoder for both subjects. The first 48 min are used as the training set for the fixed supervised decoder, explaining why there is no accuracy there. The final accuracies are computed from Session 3 onwards.

The fixed supervised decoder reaches a total accuracy of 80.2% (subject 1) and 78.5% (subject 2), while the proposed time-adaptive unsupervised decoder reaches a total accuracy of 80.2% and 83.0%. The latter thus performs on par with the former for the first subject while outperforming the former with 4.5% for the second subject. Furthermore, our approach does not require an a priori calibration session with the end-user but can be implemented in a plug-and-play fashion on a device. Lastly, one can imagine that given this is still a controlled experiment, more severe changes in the setup and conditions can occur (for instance, one of the electrodes gets loose). Our approach would be able to adapt to such changes, while the fixed decoder will only perform worse.

To evaluate whether, besides the favorable memory usage, the recursive implementation also benefits from the exponential weighting w.r.t. the uniform weighting in a sliding window implementation in a time-adaptive context, we test the sliding window implementation with $K = 19$ (i.e., equivalent to $\alpha = \beta = 0.9$) but without repredictions of the labels (to be in line with the recursive implementation which also does not repredict previous labels). The resulting accuracy is 79.9% (subject 1) and 81.6% (subject 2). While this is for both subjects worse than the recursive implementation, we cannot draw firm conclusions about this based on two subjects alone. However, as expected, these results at least suggest that an exponential weighting is favorable w.r.t. a uniform weighting in a time-adaptive context.

Although data from only two subjects are available in this experiment, hampering clear statistical conclusions, the results clearly show the potential of the proposed time-adaptive unsupervised decoder in a practical AAD use case.

VI. Discussions and Conclusions

We adapted the offline batch version of the unsupervised stimulus reconstruction algorithm for AAD as proposed in [8] to a time-adaptive online version. This allows the decoder to automatically adapt to non-stationarities in the EEG and audio. We have developed both a sliding window implementation with repredictions of previous labels in a finite pool and a single-shot predict-and-update recursive implementation without repredictions. The latter has the advantage, as it results in similar performances (Section IV) for much less memory usage and computational requirements. Via extensive experiments, we have selected the algorithm's hyperparameters and validated this on an independent dataset. Furthermore, we explained why there are hardly any changes in labels when using iterative repredictions in this time-adaptive context, while this is the case in the batch-mode algorithm presented in [8].

Lastly, the proposed time-adaptive unsupervised decoder outperformed the state-of-the-art fixed supervised decoder on a dataset that reflects a practical AAD use case (with testing across multiple sessions on different days). Given that this dataset only contains two subjects, we are careful in drawing firm conclusions. The results, however, clearly show the potential of the proposed method.

As explained in [5], [8], the stimulus reconstruction method does not perform well enough on short decision segment lengths for the online AAD application. In Section V, there were no switches in auditory attention present, such that a high accuracy could be obtained with 30 s decision segments. While this reduces the relevance of the proposed algorithm as the 'decision-maker' in AAD, it is an excellent candidate to provide reliable labels of the auditory attention to update a faster algorithm (such as the common spatial pattern filtering algorithm [19]) in an unsupervised manner but only at low speeds. As an automatic 'labeler' to inform another (supervised) AAD algorithm, the speed at which labels are produced is less restricted.

In conclusion, the developed time-adaptive unsupervised

stimulus reconstruction method is an important step forward to the online application of AAD in neuro-steered hearing devices.

REFERENCES

[1] S. J. Aiken and T. W. Picton, "Human Cortical Responses to the Speech Envelope," *Ear and Hearing*, vol. 29, no. 2, pp. 139–157, 2008.

[2] N. Mesgarani and E. F. Chang, "Selective cortical representation of attended speaker in multi-talker speech perception," *Nature*, vol. 485, pp. 233–236, 2012.

[3] N. Ding and J. Z. Simon, "Emergence of neural encoding of auditory objects while listening to competing speakers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. 29, pp. 11 854–11 859, 2012.

[4] S. Van Eyndhoven, T. Francart, and A. Bertrand, "EEG-Informed Attended Speaker Extraction from Recorded Speech Mixtures with Application in Neuro-Steered Hearing Prostheses," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 5, pp. 1045–1056, 2017.

[5] S. Geirnaert, S. Vandecappelle, E. Alickovic, A. de Cheveigné, E. C. Lalor, B. T. Meyer, S. Miran, T. Francart, and A. Bertrand, "Electroencephalography-Based Auditory Attention Decoding: Toward Neurosteered Hearing Devices," *IEEE Signal Processing Magazine*, vol. 38, no. 4, pp. 89–102, 2021.

[6] J. A. O'Sullivan, A. J. Power, N. Mesgarani, S. Rajaram, J. J. Foxe, B. G. Shinn-Cunningham, M. Slaney, S. Shamma, and E. C. Lalor, "Attentional Selection in a Cocktail Party Environment Can Be Decoded from Single-Trial EEG," *Cerebral Cortex*, vol. 25, no. 7, pp. 1697–1706, 2014.

[7] M. J. Crosse, G. M. Di Liberto, A. Bednar, and E. C. Lalor, "The Multivariate Temporal Response Function (mTRF) Toolbox: A MATLAB Toolbox for Relating Neural Signals to Continuous Stimuli," *Frontiers in Human Neuroscience*, vol. 10, p. 604, 2016.

[8] S. Geirnaert, T. Francart, and A. Bertrand, "Unsupervised Self-Adaptive Auditory Attention Decoding," *IEEE Journal on Biomedical and Health Informatics*, vol. 25, no. 10, pp. 3955–3966, 2021.

[9] W. Biesmans, N. Das, T. Francart, and A. Bertrand, "Auditory-Inspired Speech Envelope Extraction Methods for Improved EEG-Based Auditory Attention Detection in a Cocktail Party Scenario," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 5, pp. 402–412, 2017.

[10] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365–411, 2004.

[11] Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero, "Shrinkage algorithms for MMSE covariance estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5016–5029, 2010.

[12] S. Geirnaert, T. Francart, and A. Bertrand, "An Interpretable Performance Metric for Auditory Attention Decoding Algorithms in a Context of Neuro-Steered Gain Control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 1, pp. 307–317, 2020.

[13] S. J. Orfanidis, "Exponential Smoothing," in *Applied Optimum Signal Processing*, 2018, pp. 221–315.

[14] N. Das *et al.*, "Auditory Attention Detection Dataset KULeuven," Zenodo, 2019. [Online]. Available: https://zenodo.org/record/3997352

[15] S. A. Fuglsang, T. Dau, and J. Hjortkjær, "Noise-robust cortical tracking of attended speech in real-world acoustic scenes," *NeuroImage*, vol. 156, pp. 435–444, 2017.

[16] S. A. Fuglsang, D. D. E. Wong, and J. Hjortkjær, "EEG and audio dataset for auditory attention decoding," *Zenodo*, 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1199011

[17] K. Ogata, "Transient and Steady-State Response Analyses," in *Modern Control Engineering*, 5th ed. Pearson, 2010, pp. 159–268.

[18] R. Zink, S. Proesmans, A. Bertrand, S. Van Huffel, and M. De Vos, "Online detection of auditory attention with mobile EEG: closing the loop with neurofeedback," *bioRxiv*, no. 218727, 2017.

[19] S. Geirnaert, T. Francart, and A. Bertrand, "Fast EEG-based decoding of the directional focus of auditory attention using common spatial patterns," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 5, pp. 1557–1568, 2021.