Active inference as a unifying, generic and adaptive framework for a P300-based BCI

Jelena Mladenovic, Jérémy Frey, Mateus Joffily, Emmanuel Maby, Fabien Lotte, Jérémie Mattout

Active Inference as a Unifying, Generic and Adaptive Framework for a P300-based BCI

Jelena Mladenovic[1,2], Jeremy Frey[3], Mateus Joffily[4], Emmanuel Maby[2], Fabien Lotte[1] and Jeremie Mattout[2]

[1] Potioc team, Inria Sud-Ouest, Bordeaux, France; [2] Ullo, La Rochelle, France; [3] Cophy team, CRNL, INSERM U1028, Lyon, France; [4] Univ Lyon, CNRS, GATE UMR5824, F-69130, Ecully, France;

E-mail: jelena.mladenovic@inria.fr

Abstract.

Objective. Going adaptive is a major challenge for the field of Brain-Computer Interface (BCI). This entails a machine that optimally articulates inference about the user's intentions and its own actions. Adaptation can operate over several dimensions which calls for a generic and flexible framework. *Approach.* We appeal to one of the most comprehensive computational approach to brain (adaptive) functions: the Active Inference (AI) framework. It entails an explicit (probabilistic) model of the user that the machine interacts with, here involved in a P300-spelling task. This takes the form of a discrete input-output state-space model establishing the link between the machine's (i) observations – a P300 or Error Potential for instance, (ii) representations – of the user intentions to spell or pause, and (iii) actions – to flash, spell or switch-off the application. *Main results.* Using simulations with real EEG data from 18 subjects, results demonstrate the ability of AI to yield a significant increase in bit rate (17%) over state-of-the-art approaches, such as *dynamic stopping*. *Significance.* Thanks to its flexibility, this one model enables to implement optimal (dynamic) stopping but also optimal flashing (i.e. active sampling), automated error correction, and switching off when the user does not look at the screen anymore. Importantly, this approach enables the machine to flexibly arbitrate between all these possible actions. We demonstrate AI as a unifying and generic framework to implement a flexible interaction in a given BCI context.

1. Introduction

A Brain-Computer Interface (BCI) is a system that instantiates a direct interaction with the brain, be it (i) for restoring control (e.g. for movement [1] or communication [2]); (ii) for assistance and task optimization (e.g. by monitoring workload) [3] or (iii) for rehabilitation by enabling the self-regulation of brain activity for therapeutic purposes (Neurofeedback) [4].

One of the most commonly used non-invasive BCI for communication is the visual P300-speller [2]. It relies on event-related potentials (ERPs) notably including the P300 – an EEG positive deflection occurring around 300ms after a rare and relevant event. This event can be the display or highlighting of an expected item (e.g. a letter, a number or a picture). With a P300-speller, the subjects are typically presented with a 6×6 grid of characters, where a set of

items within a row or a column are flashed in a pseudo-random order (the Row-Column – RC paradigm). To select a letter, during the flashing, the users need to focus their visual attention on the item they wish to spell. Once the target item is flashed, the brain reacts with a P300, enabling the machine to detect the particular ERP and spell the desired character. Online, the machine aims at inferring which stimulus corresponds to the targeted item. In order to gain confidence about the target letter, the machine flashes the items in repetition. Intuitively, one would believe that the longer the machine flashes, the higher the confidence. However, this is not necessarily the case, as the user's vigilance may drop over time which affects the EEG signals and hence classification accuracy. For more details see [5].

1.1. Related Work

Although the P300-speller has a relatively high Information Transfer Rate (ITR) compared to other BCIs, it remains a fairly slow and cumbersome mean of communication due to the necessity of trial repetition for a fairly correct P300 classification [6]. In our context, it is interesting to consider such improvements as belonging to either one or the other of the two following categories:

(1) Static methods, that implement static design enhancements to increase the signal-to-noise ratio (SNR), e.g. by (i) preventing perceptual errors such as the "repetition blindness" – when flashing the same item consecutively [7, 8], or the "near-target effect" – when flashing within a close range both temporally and spatially from the target letter [9], varying the inter-stimulus intervals or flashing patterns [5]; or (ii) motivating users with more engaging playful environments [10], captivating stimuli (smileys [8] or real faces [8, 11]), intelligent (but not data-dependent) order of stimuli apparition [12, 13]; inter-symbol distance, symbol size, contrasted foreground and background colours [14] or monetary rewards [15].

(2) Dynamic methods, that endow the machine with flexibility or adaptive behavior such that it will adjust some of its design parameters based on the online acquired signals and the states of the ongoing interaction. These usually include probabilistic or Bayesian approaches to update the machine's belief in real time and optimize the resulting decisions. For instance, optimal (or dynamic) stopping both reduces the number of flashes and increases accuracy by using the brain response to each flash to update both its belief about the target letter and its confidence about this belief [16, 17].

In [17], the outcome of a probabilistic classifier is updated online, permitting the machine to stop and spell a letter once it attains a predefined confidence level. Here the decision speed (number of flashes) depends on the consistency and reliability of accumulated evidence. Another example is the effort to get rid of individual calibration, by implementing unsupervised classification [18], or by adopting a transfer learning strategy based on data from previous subjects [19]. To go further in assisting the user to spell words, some authors implement language models together with the optimal stopping to reach higher ITR [20]. Automatic spelling corrections using Error Potentials (ErrPs) have also been used [21, 22]. It should be noted that the subject directly influences the level of improvement that can be achieved. Indeed, when users reach higher accuracy thanks to adaptive machines, they become

more motivated, which in turn yield higher SNRs hence an even higher accuracy. A virtuous cycle that has been evidenced online when implementing optimal stopping [17]. And most recent advances in adaptive P300 spellers go beyond optimal stopping by also incorporating optimal flashing, a form of active sampling that consists in flashing the group of letters that should provide most information to reveal the target [23].

Considering (1), some "static methods" could apply to every subject (such as prevention of near target or repetition blindness effects), but other solutions, such as different colours, letter size, inter-stimulus intervals, or 3D environments seem to be non-transferable across all subjects. Typically, those are specific to a particular BCI scenario, person or even time period. Furthermore, these methods are not sensitive to changes in user states (they do not adapt), for instance they could not account for user fatigue. We believe that these static solutions can increase the initial usability, but not a long-lasting one. We find it is of essential importance to merge the knowledge used for static design methods and apply it in a dynamic way.

Considering (2), the "dynamics methods" – the few existing adaptive developments have been designed independently of each other, namely, adaptive flashing and adaptive spelling. It appears difficult to combine such adaptive actions in one computational framework, as one needs to find a way for the machine to optimally arbitrate, online, between alternative actions. For instance, in adaptive stimulus presentation as proposed by [23], the authors used a probabilistic model to implement optimal stopping with a fixed decision threshold, and relative entropy with a greedy search algorithm to select the next sequence of flashes. However, such a solution is not generic in the sense that the action space remains limited and specific to the particular phase of the ongoing interaction (e.g. flashing, spelling or correcting). As a consequence, right after spelling an item for instance, the machine cannot choose between validating this item or flashing again to acquire more evidence, or immediately spelling another item instead. Furthermore, as such a decision relies on the ability to detect an Error Potential (ErrP), one has to be able to evaluate the confidence of ErrP detection within a single trial, which is a very noisy step. As a matter of fact, ErrP classifiers have to be used online with precaution. This is because in case of low specificity (i.e. high risk of labeling a correct letter as an error), the correct letters can be replaced with another (wrong) one. This phenomenon has shown to be quite frustrating for users [21]. Some authors even recommend not to use such corrections, stating that word auto-completions using contextual and language models suffice [24]. Indeed, for such an automated correction to be effective, an adaptive framework is needed to optimally weight all possible alternative decisions, based on their relative predictions and confidence. In particular, this requires a unifying framework in which the various relevant quantities can be negotiated in a common currency. For instance, additional information need to be traded with the time needed to get that information and, as well, with the expected reward associated with error-free communication.

1.2. A unifying framework

The required unifying framework puts an emphasis on the various decisions and actions the machine may take. In that sense, it extends the adaptive approaches that implement learning

abilities only (e.g. adaptation of feature extraction or classification parameters over time using machine learning techniques) with active sampling which provides actions in a way that also influences the user and optimizes the interaction. We have previously advocated for these two complementary aspects of adaptation in BCI and proposed a unifying conceptual framework in [25].

In this paper, we propose and illustrate an instantiation of the conceptual framework for adaptive BCIs from [25], based on a recent computational model developed in theoretical neuroscience and called Active Inference [26]. It resides on the mentioned perception-action cycles that couple the agent to its environment. Note that in our context, the environment of interest for the machine or (artificial) Active Inference agent is the BCI user. Active Inference rests on a generic Bayesian approach that we show could incorporate various instances of adaptive BCI techniques into one flexible framework. It involves a formal generative model, in which the dependencies between observations, user states (intentions) and actions are specified given a particular context (here a P300-speller BCI). Based on this probabilistic model and an optimization criterion referred to as the Free Energy Principle (FEP), the machine infers the user's intentions (what letter to spell, if none then pause) from EEG observations and computes optimal actions (to flash or spell). Applying Active Inference in a P300 speller context thus naturally endows the interaction with *optimal flashing and spelling*. Importantly, Active Inference turns an optimization problem (action selection) into a Bayesian inference one where preferences or goals are specified in the form of prior expectations. Desired outcomes are encoded in terms of quantitative priors.

We apply Active Inference on a simulated P300-speller, using real data from 18 subjects. Moreover, to demonstrate the flexibility of this framework, we implement various adaptive features such as automated error correction or the detection of a state where the user is looking away from the screen. As these features correspond to alternative (hidden) states that the machine's model of the user considers plausible, and since the Active Inference framework rests on a single optimization criterion (the FEP), the machine will automatically arbitrate between all possible actions based on both in-build priors and incoming observations. Note that in this first demonstration of Active Inference for BCI, we consider a simplified situation where observations are not raw EEG data but appropriately pre-processed, extracted and classified features. In other words, the Active Inference framework is here plugged-in on top of a classical feature extraction and (probabilistic) classifier for P300-based BCI.

In the following sections, we first summarize the general principle of Active Inference in (2.1), emphasizing its genericity and flexibility. We describe in (2.2) how Active Inference can be applied in the context of P300-speller BCI. In (3), wee introduce the real data and features we used to evaluate this new approach by simulating online spelling. The following section (4) contains the obtained results, comparing Active Inference with state-of-the art algorithms. Finally, (5) includes a discussion, and (6) comprises our concluding words and future work.

2. Methods

The Active Inference framework has been proposed as a biologically plausible computational model of the brain [26]. Here we build on the analogy between the brain and any adaptive system. We endow the machine with Active Inference in order to enable it to flexibly interact with the user in a P300-based BCI. In the following subsection, we introduce Active Inference as proposed in computational neuroscience, and draw the brain-machine analogy.

2.1. Active Inference

By the end of the last century, neuroscientists ceased to perceive the brain as a passive organ which simply processes stimuli, but as an active organ that constantly updates a (probabilistic) model of its environment and predicts future sensory inputs [27]. This view has given rise to the so-called Bayesian brain hypothesis whereby the brain is thought to implement (approximate) Bayesian inference. A compelling computational framework that incorporates the Bayesian brain hypothesis aiming at explaining perception, learning and decision making in biologically plausible terms. In this scope, the most advanced General Framework both computationally and theoretically is Active Inference [26]. It extends approximate Bayesian Inference by tightly coupling perception with action (unifying cause and effect). In other words, as living beings cannot directly perceive the true states of the world (the cause), they need to infer them from noisy observations (the effect). Such inference is achieved by repeatedly performing perception-action cycles. They constantly anticipate the true states and represent them within a generative model of sensory inputs. This way they are *implicit Bayesian modelers of their environment* [26] In order to exchange with an ever-changing environment and maintain homeostasis, biological (adaptive) systems restrict themselves to a limited number of states. In other words they are resisting the natural tendency of dispersion (resisting the 2^{nd} law of thermodynamics) [26]. This mechanism can be seen as minimizing the entropy (disorder or unpredictability) of the distribution over the outcomes they experience (observations) relative to a desired outcome (e.g. homeostasis).

2.1.1. A Brain - Machine Analogy In BCI, the observations (EEG) are often very noisy and contain high variability for which we often do not know the cause, and thus cannot control its outcomes to our favor. We wish to endow the machine with Active Inference, in order to model the causes of observations, to better anticipate and favor certain outcomes. This is indeed what we are looking for in BCI systems. As such, let us draw a parallel between (i) **the brain** as an adaptive system, described by Active Inference, which:

- accumulates observations to update its internal model of the environment,
- optimizes its interactions through making inference about the environment,
- optimizes its interactions through acting upon the environment;

and (ii) **the machine** which should incorporate the same behavior to achieve an adaptive BCI, namely:

- accumulate observations – EEG data – to update its internal model of the user, e.g., the model containing probabilities of user's intentions, states, reactions to machine's actions etc.
- optimize its interactions through making inference about the user, i.e., with the updated user model, updated prediction for a certain user state e.g., fatigue or intention to spell or pause
- optimize its interactions through acting on the user, i.e., with the updated user model, reinforce predictions or reduce prediction error with optimal action (feedback or stimuli).

Both the brain and the machine behave in order to best anticipate future outcomes by minimizing entropy (minimizing chaos, or maximizing information) relative to a desired outcome. In the following, we expose (a) the generic discrete state space model used by the Active Inference framework to model sequential learning and decision making by the brain, and (b) the objective function (relative entropy) it minimizes – free energy.

2.1.2. Generative Bayesian model Sensory evidence (observation) is evaluated and updated given a generative model m under Markovian assumptions in order to reach optimal predictions. The model contains priors over future outcomes that encode one's goals or preferences. Such priors influence action selection, as depicted in Figure 1. Note that m embeds the generative model assumptions specific to each agent.

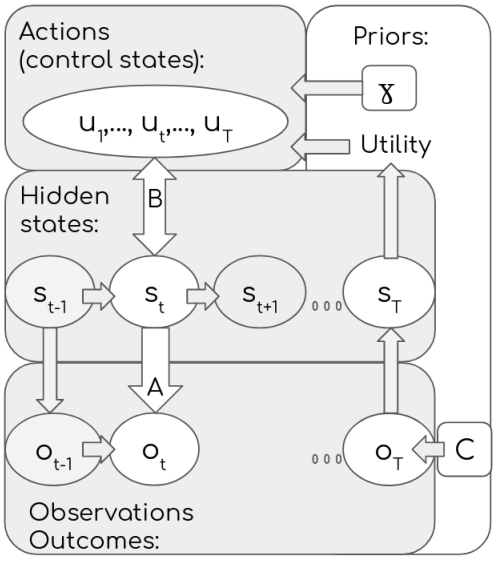

Figure 1: Illustrates Markov model of hidden states S, control states U and observations O. Actions are sampled from the posterior probability distribution over control states, which is parametrized by the precision parameter γ and preferences over future outcomes C. The latter assigns high values to desired final outcomes or states and penalizes undesired ones.

The generative model m is a joint probability over hidden states $\mathbb{S}$, control states $\mathbb{U}$, observations $\mathbb{O}$ and model parameters:

$\mathbb{S}$ **– finite set of hidden states:** Hidden states are internal representations a living being (or a machine in our case) can have about the hidden causes of their sensations (observations). For instance, they can be the letter on the user's mind that cause a P300 EEG deflection (machine's observation) after the presentation of flashing letters (machine's action).

$$\mathbb{S} = s^{(1)}, s^{(2)}, .., s^{(n)}, \text{ with } |\mathbb{S}| = n;$$

Let s map each trial t onto one element from finite set $\mathbb{S}$;

$$s(t) = s_t \in \mathbb{S}, \quad \forall t = 1, ..., T$$

where n represents the number of possible states, or cardinality of $\mathbb{S}$ at every trial t; T is the final trial, and t the current one. This means that only one state out of n possible ones can take place at a time or trial t.

$\mathbb{U}$ **– finite set of control states or actions:** In active inference, actions are sampled from beliefs about control and, thus need to be inferred from observations. However for simplicity, in most implementations of active inference, realized actions are assumed to be known by the agent. The agent entertains posterior beliefs about the control of (hidden) state transitions. In the previous example, a possible action would be the flashing of a specific letter.

$$\mathbb{U} = u^{(1)}, u^{(2)}, .., u^{(r)}, \quad \text{with } |\mathbb{U}| = r;$$

Let u map each trial t onto one element from finite set U;

$$u(t) = u_t \in \mathbb{U}, \quad \forall t = 1, ..., T$$

where r represents the number of possible states, or cardinality of $\mathbb{U}$ at every trial t. Only one action out of r possible ones can take place at a time or trial t.

$\mathbb{O}$ **– finite set of observations or outcomes:** Observations are anything an agent can directly sense. In our example, taking the machine's perspective, they are the (discrete) EEG signal.

$$\mathbb{O} = o^{(1)}, o^{(2)}, .., o^{(z)}, , \text{ with } |\mathbb{O}| = z;$$

Let o map each trial t onto one element from finite set $\mathbb{O}$;

$$o(t) = o_t \in \mathbb{O}, \quad \forall t = 1, ..., T$$

where z represents the number of possible observations, or cardinality of $\mathbb{O}$ at every trial t. Only one observation out of z possible ones can take place at a time or trial t.

The generative (Bayesian) model as defined in [28] writes:

$$P(\tilde{o},\tilde{u},\tilde{s},\gamma|m) = \underbrace{P(\tilde{o}|\tilde{s},m)}_{likelihood}\underbrace{P(\tilde{s},\tilde{u}|\gamma,m)}_{transitions}\underbrace{P(\gamma|m)}_{precision} \tag{1}$$

where $\tilde{o} = o_1,..,o_T \in \mathbb{O}$, $\tilde{s} = s_1,..,s_T \in \mathbb{S}$, $\tilde{u} = u_1,..,u_T \in \mathbb{U}$. Note that we denote matrices with bold capital letters and vectors with only capital letters. The model is defined by three major elements, as given in equation (1):

(i) **Likelihood matrix A**, from (1): represents the likelihood of observations given the hidden states:

$$P(\tilde{o}|\tilde{s},m) = \prod_{i=1}^{T}\underbrace{P(o_i|s_i,m)}_{likelihood}, \quad P(o_i = k|s_i = h) = \mathbf{A}_{k,h}$$

where $\mathbf{A} \in \mathbb{R}^{z\times n}$. In other words, given each $h = 1,..n$ states there is a probability to get a $k = 1,..z$ observation. Thanks to the likelihood, our Bayesian model contains probabilities from the past experience, and enables us to predict the probability to perceive a new observation o_{t+1} given a state s_{t+1}.

(ii) **Probabilistic transition matrix between states B(u_t)**, given an action, from (1):

$$P(\tilde{s},\tilde{u}\,|\,\gamma,m) = P(u_t|\gamma,m)\prod_{i=1}^{t}\underbrace{P(s_{i+1}|s_i,u_i,m)}_{transitions}; P(s_{t+1} = w \mid s_t = q,u_t) = \mathbf{B}(u_t)_{w,q}$$

where $w,q = 1,..n$, hence $\mathbf{B}(u_t) \in \mathbb{R}^{n\times n}$, and n refer to the number of hidden states.
This means that transitions between hidden states depend upon the current *putative* action u_t under policy $\pi \in 1,\cdots,K$. A policy indexes a specific sequence of control states $(\tilde{u}|\pi) = (u_t,\cdots,u_\tau|\pi)$:

$$\begin{aligned} \ln P(\tilde{u}|\gamma,m) &= \underbrace{\gamma}_{precision}\cdot\mathbf{Q}(\pi) = \gamma\cdot(\underbrace{\mathbf{Q}(u_{t+1}|\pi)+\cdots+\mathbf{Q}(u_\tau|\pi)}_{expected(negative)\ free\ energy}) \\ \mathbf{Q}(u_\tau|\pi) &= \underbrace{E_{Q(o_\tau|\pi)}[\ln P(o_\tau|m)]}_{extrinsic\ value} + \underbrace{E_{Q(o_\tau|\pi)}[D_{KL}[Q(s_\tau|o_\tau,\pi)|Q(s_\tau|\pi)]]}_{epistemic\ value} \end{aligned} \tag{2}$$

weighted by the precision parameter γ (detailed bellow in 2.1.2), such control states or *putative* actions are chosen to minimize expected free energy, where D_{KL} is the Kullback-Leibler (KL) divergence or relative entropy (for more on KL divergence, see Appendix 1); and $E_{Q(o_\tau|\pi)}$ is the expectation of a future outcome o_τ given policy π. For the sake of readability we develop each element from equation (2), as follows.

An action u_t is chosen from a list of putative actions u_τ under a given policy π that minimizes expected fee energy which is comprised of 2 elements:

(i) Extrinsic value or the preferred final outcome (the goal we wish to achieve) which we maximize, that is its expectation $E_{Q(o_\tau|\pi)}$.

(ii) Epistemic value or information which we wish to maximize, that is, its expectation $E_{Q(o_\tau|\pi)}$. That is equivalent to minimizing the prediction error, or the discrepancy between the prior (predicted hidden state or prior $Q(s_\tau|\pi)$) and posterior (actual hidden state given the observation $Q(s_\tau|o_\tau,\pi)$). We achieve this by minimizing the relative entropy (i.e., minimizing the KL divergence relative to the predicted outcome).

You can notice that $E_{Q(o_\tau|\pi)}$ of a probability distribution Q (called the variational) is used twice, and serves as a bound and link between different probability distributions P and Q, that describe the *extrinsic value* and *epistemic value,* respectively (for more details, see Appendix 2).

So, we are wagering between the epistemic and extrinsic value at each iteration, i.e., trying to get closer to the prior goal (future outcome) by acquiring maximum information.

The extrinsic value contains $P(o_\tau|m)$, which is the prior distribution over future outcomes, referred to as $\mathbf{C}_\tau$. So, let $\mathbf{C}_\tau$ be the preference of future outcomes $o_\tau \in \mathbb{O}$. As part of *extrinsic value*, it influences the choice of action to reach such desired outcomes. If we consider all available observations from set $\mathbb{O}$ as future outcomes then $o_\tau = o^{(z)}$:

$$\mathbf{C}_\tau = \sigma([C(o^{(1)}),C(o^{(2)}),..,C(o^{(z)})])^T$$

where σ is a *softmax* (normalized exponential function) of final outcomes, such that:

$$\begin{gathered}\sigma : \mathbb{R}^z \mapsto \mathbb{R}^z, \\ \sigma(o_\tau)_j = \frac{e^{(o\tau)_j}}{\sum_{i=1}^{z} e^{(o\tau)_i}} \in (0,1), \ \ \forall j = 1,.,z\end{gathered} \tag{3}$$

The *softmax* function here compresses a z-dimensional vector $[C(o^{(1)}),C(o^{(2)}),..,C(o^{(z)})]$ of real values into another $\mathbf{C}_\tau$ vector of the same dimension that contains real values that add up to *1* and reside within the range of *(0,1)*. In other words, we transform all observations from set $\mathbb{O}$ into prior probabilities of future outcomes, some of which we favor, other which we penalize.

(iii) **Precision parameter** γ, from (1) :

$$P(\gamma|m) = \Gamma(\alpha,\beta)$$

where Γ is a gamma distribution of scale parameter α and rate parameter β. If a random variable X follows a Gamma distribution then:

$$f(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \text{ for } x > 0 \text{ and } \alpha,\beta > 0$$

where $\Gamma(\alpha)$ is the gamma function (i.e. an extension of the factorial function):

$$\Gamma(\alpha) = (\alpha-1)!$$

The precision parameter (also called temperature) determines the degree of confidence of the control states or beliefs over actions. For example, if $\gamma \mapsto \infty$ the beliefs over policies merge into a single policy, being over optimistic and prone to errors, with immediate or fast

action (increased exploitation), inversely if $\gamma \mapsto 0+$ the beliefs over policies spread uniformly resulting as a very high exploration or waiting time. In short, the higher the confidence about having a good policy (i.e. belief of high precision), the smaller the exploration and vice versa.

We have detailed the components of the internal, Bayesian, generative model, a distribution $P(o_t, s_t, u_t, \gamma | m)$ that connects observations o_t to hidden states s_t through control states u_t.

"*The agent and the environment interact in cycles. In each cycle, the agent first figures out which hidden states are most likely by optimizing its expectations with respect to the free energy of observations. After optimizing its posterior beliefs, an action is sampled from the posterior marginal over control states. The environment then picks up this action, generates a new observation and a new cycle begins*". [29]

2.2. Active Inference for the P300-speller

We aim at designing a fully adaptive P300 speller that learns and acts optimally in real time. The above generic and flexible probabilistic framework, Active Inference enables the machine to automatically and optimally update an internal model of the environment (here the user given a BCI task) and select appropriate actions. Specifically for the P300-speller, the actions to be considered are – flashing or spelling letters or switching off the screen. This allows us to implement within the same framework: (1) optimal stopping & flashing but also when (2) the user is looking away from the screen – "lookAway" case, in which the machine can pause the application; together with above mentioned, we can also implement (3) an ErrP classifier, where after receiving an ErrP, the machine can automatically choose to spell the next probable letter or continue flashing to increase evidence for the target letter.

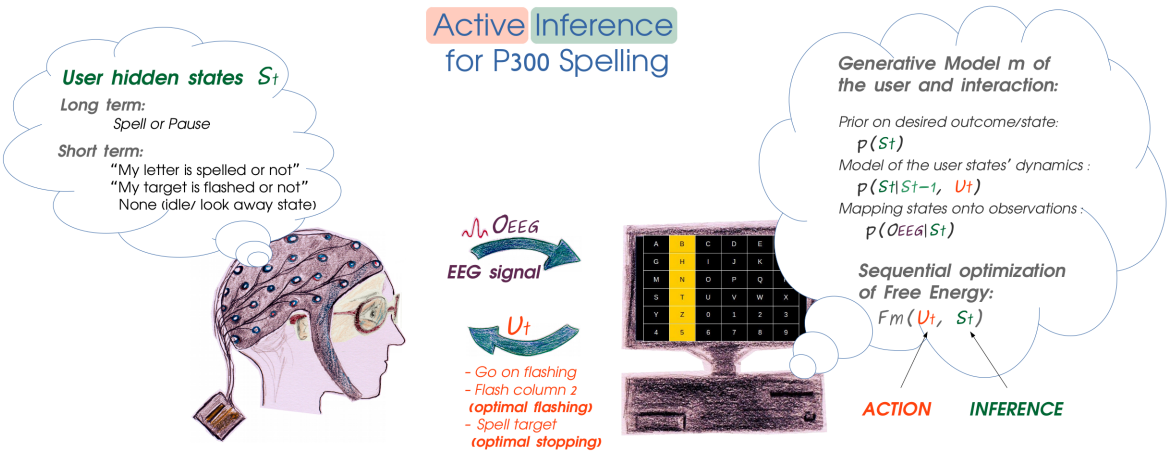

Figure 2: A depiction of Active Inference for a P300 speller: (1) the user hidden states on the left represented as long term intentions and short term reactions to stimuli, (2) the observations are the (preprocessed) EEG signal, (3) the actions the machine based on its internal (generative) model of the user. The generative model m is simplified in this figure, representing Free Energy as a function of hidden states (updated with observations) and actions $F_m(u_t, s_t)$.

When endowing the machine with Active Inference, in a P300 speller application (see Figure 2), the machine:

- accumulates the information about the target/non target letters (P300 or not) and incorrectly spelled letters or not (ErrP or not), to update its belief about the user's intention or command,
- optimizes its interactions through inference, i.e., minimizes discrepancy between observed data and predictions about user intentions to spell a letter, or pause;
- optimizes its interactions with the user by spelling and flashing items or switching-off in a flexible and adaptive manner, in order to maximize speed and accuracy.

In the next two sections, we explicitly describe the key model parts when instantiating the P300 speller BCI within the Active Inference framework. We start first with the machine's generative, internal model of the user in section 2.3, and then describe its possible actions towards the user in section 2.4.

2.3. Generative model of the user.

Prior to any observation and in the absence of prior knowledge, the probability of the intention to spell a given letter is the same for all the letters (high entropy). Then, after each flash and electrophysiological observation, these beliefs are updated based on the generative model m which embodies the machine's internal representation of the user and task.
The model m rests on transitions among hidden states that are coupled with actions, in our example it contains:

$\mathbb{S}$ – finite set of user hidden states:
There are 37 *intentions* $\times$ 4 *reactions* = 148 possible user hidden states the machine must infer. The first are the user's *intentions to spell* one out of 36 letters or digits at a time, within the 6×6 grid, or the 37^{th} intent to pause by looking away from the screen. Such state we refer to as a *lookAway* state and it enables asynchronous BCI behavior. The second represent 4 user's *reactions* to the machine's actions or stimulations. Namely, user intentions are inferred by the machine through an accumulation of short term user's reactions to stimuli being – *"My target letter was just flashed"* – giving a P300 (target)observation, or inversely – *"My target was not flashed"* – yielding a non-P300 (non-target) observation. Another type of user reaction is *"My target letter was spelled"* – or – *"What is spelled is not correct"* – giving rise to an Error Potential (ErrP) as observation. Active Inference enables us to infer the cause of sensory observations, here the user intentions, which in turn are influenced by the machine's actions.

$\mathbb{U}$ – finite set of machine control states or actions:

There are 36 *spelling* + 12 *flashing* + 1 *switch-off* = 49 possible machine's actions that can help the machine learn about the user hidden states and accomplish the user's goal. There can be 12 possible *flashing* (6 columns and 6 rows) without repetition, or *spelling* one out of 36 letters; or *switching-off* the screen in the case of a "lookAway" state.

$\mathbb{O}$ **– finite set of observations or outcomes**: Active Inference instantiated in [28] deals with discrete observations, namely in our case : (1) high or low confidence discrete values associated with the observation of a target or non target signal, and similarly (2) high or low confidence values associated with the observation of a correct or incorrect feedback. This means that after each flash, the machine observes either target (P300) or non target values with a certain degree of confidence. Similarly after each spelling the machine observes either an correct or incorrect (ErrP) feedback with more or less confidence. These confidence levels are given by the class probabilities estimated by the classifier. We denote them as follows: for a correctly spelled letter, we refer to as a Feedback Correct FC (FC0, FC1 for not confident and for confident correct feedback, respectively); and Feedback Incorrect as FI (FI0, FI1 not confident and confident incorrect feedback, respectively). If the machine is completely unsure whether the feedback is correct or not, it is classified as undefined feedback, or FXX. Same applies to flash target and non target ($T0$, $T1$ and NT0, NT1 for not confident and confident target, and non target, respectively) and TXX for an undefined response to a flash, as depicted in Figure 3.

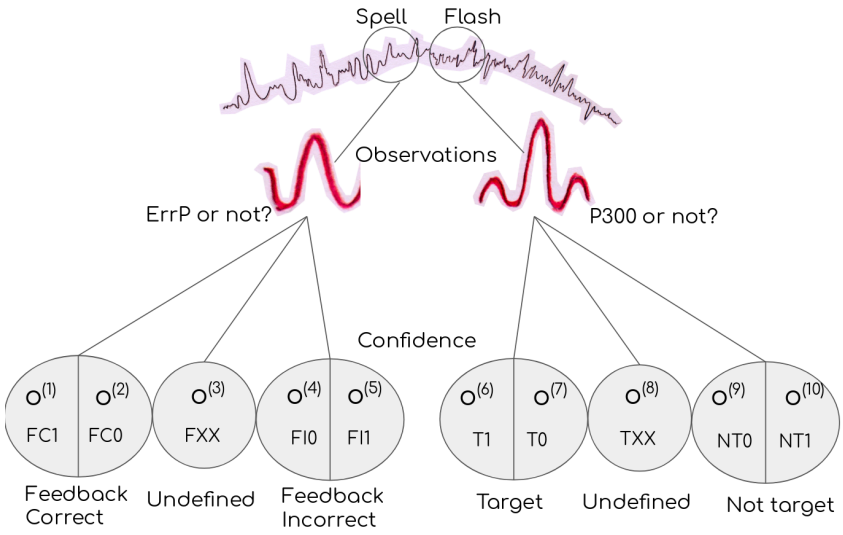

Figure 3: After each flash or spell, an observation – target/non-target or feedback correct/incorrect – is being mapped to a discrete high-to-low degree of confidence or undefined observation $O^{(i)}$, i=1..10. Those discretized observations are the ones that enter the Active Inference model.

***A* – prior over outcomes given a state (likelihood)**

The likelihood is the probability to observe an outcome o_t, given a state s_t, and $\boldsymbol{A}$ is a matrix of z possible observations, given n possible hidden states:

$$A = \begin{bmatrix} o^{(1,1)}, & o^{(1,2)}, & \dots & o^{(1,n)} \\ \vdots & & \dots & \dots \\ o^{(z,1)}, & \dots & \dots & o^{(z,n)}z \end{bmatrix}$$

For instance, $\boldsymbol{A}$ contains the probability to observe a high confidence target – T1 or low confidence incorrect feedback – FI0, given a user hidden state – a *column* flashed or a *letter* spelled, respectively. Thanks to $\boldsymbol{A}$, the machine knows how reliable is the classification. In BCI, $\boldsymbol{A}$ may typically be defined based on calibration or training data. This means that $\mathbf{A}$ should ideally be defined for each user specifically. This is indeed important to define the levels of confidence that will drive the BCI interpretations and actions. Namely, Active Inference will rely on those levels to decide whether it should go on flashing in order to make a reliable decision, or spell with no further due. The way we define the matrix for each individual is further described in subsection 3.1.3 pertaining to the realistic simulations we performed.

$\mathbf{B}(u_t)$ – transitions between states given an action

To transition from one state s_t to another s_{t+1} is possible through action (control states). The choice of action u_t given a state s_t depends on the priors C over the desired final outcome o_τ but also on the precision over action or the exploration/exploitation ratio γ while conforming to the free energy minimization, as mentioned in (2). Concretely, transition matrix $\boldsymbol{B}$ contains all the possible combinations of states or user intentions $n \times n$, with $|S| = n$, which we define prior to the experiment. These are the same for every subject, as follows.

$$B = \begin{bmatrix} s^{(1,1)}, & s^{(1,2)}, & \dots & s^{(1,n)} \\ \vdots & & \dots & \dots \\ s^{(n,1)}, & \dots & \dots & s^{(n,n)} \end{bmatrix}$$

where $n = 148$, containing 37×4: user intentions to spell 36 letters or pause (37^{th} *lookAway*), along with short-term user reactions to stimuli (1. correct/ 2. incorrect spelling, or 3. target/ 4. non-target flashing). For all subjects, the transition matrix B is the same, and its values are either 0 or 1. Values 0 and 1 refer to implausible and plausible state transitions, respectively. For instance, when a set of items has just been flashed, the current state might be the recognition of the target, or not, and a subsequent user's state could be the recognition of a future flash or the recognition of a displayed feedback, depending on the action taken by the machine.

C – priors over final outcomes

Vector C influences the choice of action. It expresses a goal or preference in the form of a prior probability over final outcomes, with the highest probability being given to the most desired outcome. Hence, the prior beliefs encode a utility function which, in our case will favor the high confidence *Feedback Correct* 'FC1' as final observation o_τ. This amounts to aiming at the state – *My target letter was spelled* –. In our case, we assign equal values (preferences) to the appearance of target/non target observations, while penalizing incorrect spelling, and favouring correct spelling, as in Figure 4. As we tested various values for C, we provide more details in the subsection 3.1.3 Simulations.

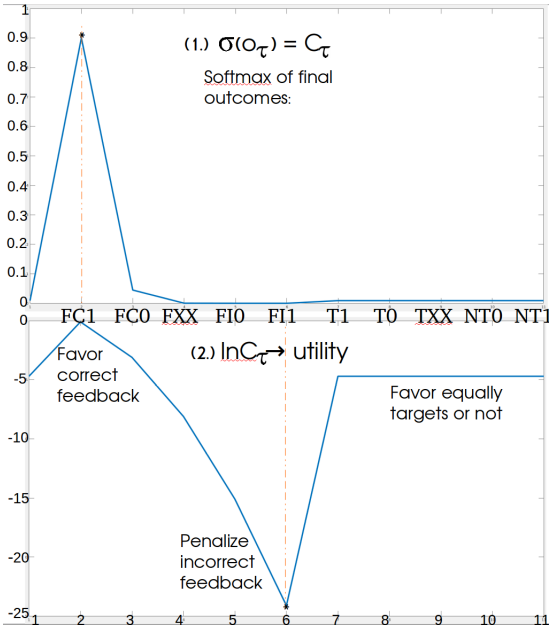

Figure 4: Upper figure: *Softmax* function yielding output values between 0 and 1 (*y-axis*) for each observation within the set $\mathbb{O}$; from FC1, FC0, FXX, FI0, FI1 which refer to feedback observations $(o^{(2)}, o^{(3)}, .., o^{(6)})$ and T1, T0, TXX, NT0 and NT1 denoting target/non-target observations $o^{(7)}, o^{(8)}, .., o^{(11)}$ (*x-axis*). Bottom figure: Logarithm of the *softmax* encodes a utility function, in which we favour the *correctly spelled letter* – FC1, and penalize *feedback incorrec*t FI1 and FI0, and *undefined* FXX feedback; while equally favouring the apparition of target T, non target NT or undefined TXX observations.

γ - precision over priors

In a P300 speller we wish to spell correct letters in a minimum amount of time. However there is always a trade-off between speed and accuracy. This trade off is governed by parameter γ which sets the balance between exploration and exploitation. In practice, this is arbitrarily set by defining the prior distribution over parameter γ (a gamma distribution with parameters α and β). As we tested multiple γ values, see 3.1.3 Simulations for more details.

2.4. Optimal Interaction

2.4.1. Optimal flashing & stopping Vector C, precision γ, and transition matrix B are defined prior to the experiment, given the task and goals. Matrix $\boldsymbol{A}$ is learned once from training data, for each user. Then, here is the course of actions that unfold during the online interaction:

- List all potential actions u_t at time t.
- Compute for each action its posterior expectation or epistemic value $E_{Q(o_\tau|\pi)}$ and compute KL divergence (also called relative entropy) between prior $Q(s_\tau|\pi)$ and posterior $Q(s_\tau|o_\tau, \pi)$ over the hidden states (using transition matrix $\boldsymbol{B}$, likelihood $\boldsymbol{A}$, preferences C and precision γ); Note that we use the full transition matrix **B** (meaning that we consider all possible hidden states during a *choice* or time t).

- The higher the information (epistemic value), the most likely this action will be chosen. When we consider not only a single putative action, but a series of actions to reach a predefined desired outcome, we then talk about policies π. Hence, we get a list of actions $(\tilde{u}|\pi) = (u_t, \cdots, u_\tau|\pi)$, active inference picks up the optimal policy, that is the one that maximizes the information gain as well as maximizes the reward (outcome).
- Update internal state s_t based on observation o_t (enabling the data-driven, adaptive model).
- Repeat the selection of the next action u_t (to flash) until the spelling of a letter (the case without an ErrP classifier); or in the case in which we use an ErrP classifier: repeat action selection (to flash or spell) until the spelling of a *correct* letter or *Feedback Correct* with high confidence FC1 (which will be obtained depending on the error rate of the feedback classifier, set in $\boldsymbol{A}$).

Active Inference permits a holistic and automatic control over the machine's actions, thanks to the free energy principle that unites action and perception (cause and effect) into a single Bayesian framework, see figure 5. As reminder, the machine chooses such action that provides most information (min entropy relative to the predefined goal or *Feedback Correct*). In that sense, flashing letters automatically provides more information about the target than spelling one by one letter.

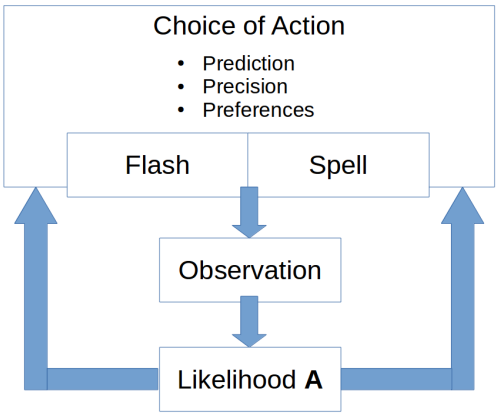

Figure 5: Simplified schematics of Active Inference choice of action. It starts by predicting the future observation or hidden state. Using priors (the precision γ and preferences C) it will choose an action to reinforce its prediction, for instance to flash a certain column; this will produce an observation (within degrees of confidence) and depending on the likelihood it will choose to continue flashing or to spell a letter. In case of an ErrP, the spelling can be followed by more flashing to reinforce its certainty about the spelled letter or immediately spell another letter.

Thanks to Active Inference, the machine is able to execute optimal flashing (with stopping), that is, flashing those letters which give most information about the target letter. Furthermore, Active Inference offers a generic and flexible framework that can also incorporate other adaptive behavioural features as described below.

2.4.2. Detecting a LookAway state We here refer to the situation where the user is not looking at the screen anymore. By simply adding another 37^{th} hidden state to the existing set $\mathbb{S}$, we provide the subject with the possibility to pause the machine. Note that there is no clearly defined single observation associated with that state which instead, can only be inferred through the absence of target like responses. In other words, if the machine observes many consecutive non target signals, it should eventually conclude that the user is not actively looking at the screen. The model thus has to be able to distinguish between a poor performing subject, producing ambiguous signals and a subject which intends to pause the P300 speller. Note that in our case we did not model a "switch on" button action, which could for instance rely on a SSVEP response with a dedicated stimulus always active at a corner of the screen. So far, we only simulate independent trials with different intentions, some of which can be a *LookAway* state to stop the machine.

2.4.3. Automated error correction We simulated an ErrP *perfect* classifier, with either a high confidence correct or incorrect feedback classifier, i.e., assigning zero probability to the appearance of *not confident* correct and incorrect feedback as well as undefined feedback, p(FC0, FI0, FXX) = 0. As this is not a very realistic case, we also simulated a more *realistic* feedback classifier, with 95% specificity (a 0.95 probability to be right about a correctly spelled letter); and 75% sensitivity (a 0.75 probability to be right about an incorrectly spelled letter). This way the confidence for specificity (Feedback Correct) is p(FC1 = 0.95; FC0 = 0; FXX = 0; FI0 = 0; FI1 = 0.05), and for sensitivity (Feedback Incorrect) it is p(FC1 = 0.25; FC0 = 0; FXX = 0; FI0 = 0; FI1 = 0.75).

If Active Inference realizes it spelled an incorrect letter, it will choose by itself to continue flashing and gain additional information about the target, or to immediately spell the second most probable letter. In the case of a *perfect* feedback classifier, it will be 100% sure about the letter whether it is incorrect or correct. In the case of the *realistic* feedback classifier, it would not be so sure (5% and 25% error for correct and incorrect letter, respectively).

In the next subsection, we describe the evaluation approach we pursued in order to validate Active Inference for implementing a flexible and efficient P300 speller BCI. This includes a description of the Dataset and Data Features, of the Model, of the Simulation procedure and of the Evaluation Metrics we used.

3. Experimental Design

3.1. Dataset

We use real training data from one of our previous studies [17] to which 18 healthy subjects (11 males and 7 females) aged from 22 to 30 took part voluntarily to evaluate the P300-speller BCI paradigm. Thirty- two EEG sensors were used and their placement followed the extended 10–20 systems. The P300-speller BCI experiment was made of two recording stages:

- the initial training phase enables to optimize spatial filters [30] and a probabilistic classifier [31] that can then be used to differentiate response-to-target data from response-

to-non-target data. In this training phase, subjects were given a sequence of 25 characters to focus on. For one character, matrix rows and columns were flashed alternatively during three complete cycles of 12 stimuli (two of which were including the target item). The training dataset is thus composed of 750 trials for the non-target class and 150 trials for the target class.

- following the training phase, each participant completed 3 copy-spelling sessions as a test phase. Each session was made of twenty-four 5-letter French words, hence 360 letters in total. The process of flashing each row and column was repeated three times (3×12) per character spelled.

3.1.1. Features From the data recorded during the test phase, the features are extracted for our simulation, as follows. A first preprocessing step consisted in applying of a 2^{nd} order bandpass Butterworth filter with cut-off frequencies of 0.5 and 20Hz.

We use Riemannian geometry, the state of the art data classification approach developed by [32]. It uses covariance matrices which are Symmetric Positive Definite (SPD) matrices and lie in a differential geometry manifold. We define such covariance matrices as follows. Let $X_i \in \mathbb{R}^{S \times N}$ the EEG epoch corresponding to N consecutive samples in response to the i^{th} stimulus recorded on S sensors, as proposed in [33] we construct the *super-trial* $\widetilde{X}_i$ with the concatenation of X_i and the temporal prototype P which is the average of all target epochs recorded during the calibration phase:

$$\tilde{X}_i = \begin{pmatrix} X_i \\ P \end{pmatrix}$$

Let us compute the corresponding covariance matrix for the i^{th} stimulus:

$$\widetilde{\Sigma}_i = \frac{1}{N-1} \widetilde{X}_i \widetilde{X}_i^T = \begin{bmatrix} \Sigma_P & C_{P,X_i}^T \\ C_{P,X_i} & \Sigma_i \end{bmatrix}$$

where Σ_i and Σ_P are respectively the covariance matrices of the X_i EEG epoch and the temporal prototype P, and $K_{X_i,P}$ the cross-covariance between the X_i EEG epoch and the temporal prototype P.

In the same way, we can compute this covariance matrix for each trial of the calibration phase for the target and non-target classes. To determine to which class (target or non target) a covariance matrix $\widetilde{X}_i$ belongs, the Riemannian distance is computed between it and the Riemannian means for target and non-target classes respectively denoted $\tilde{\Sigma}_T$ and $\tilde{\Sigma}_{NT}$, as follows. Let us consider two SPD covariance matrices K_1 and K_2, where $\|\cdot\|_F$ is the Frobenius norm, then the Riemannian distance between them is:

$$\delta_R(K_1, K_2) = \left\| (log\, K_1^{-1} K_2) \right\|_F \tag{4}$$

Knowing that the diagonal elements of such $n \times n$ covariance matrices are real positive eigenvalues λ_i, we can write the Riemannian distance as:

$$\delta_R(K_1, K_2) = \sqrt{\sum_{i=1}^{n} log^2 \lambda_i}$$

Then, for each trial X_i we can extract the following measure:

$$rTNT = \frac{\delta_R(\tilde{\Sigma}_i, \tilde{\Sigma}_T)}{\delta_R(\tilde{\Sigma}_i, \tilde{\Sigma}_{NT})}$$

For classification, we used a simple probabilistic generative model of the data, based on a two univariate-Gaussian mixture (one Gaussian distribution per class). Then, following Bayes Rule, the likelihood when seen as a conditional density can be multiplied by the prior probability density of the parameter and then normalized, to give a posterior probability density :

$$p(C_j|Y) \propto p(Y|C_j)p(C_j)$$

where Y is the feature on which the classification C_j was done, $j = 0$ referring to the target and $j = 1$ to the non-target class and with

$$p(Y|C_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{(Y-\mu_j)^2}{2\sigma_j^2}}$$

where μ_j and σ_j^2 are respectively the mean and the variance of the Gaussian distribution for the class j.

Finally, for our simulation, we calculate the log likelihood lf_j, in case the feature is the $rTNT_i$ measure for each flash, as follows :

$$lf_j = log(p(rTNT_i|C_j)) = -log(2\pi.\sigma_{rTNT_j}) - \frac{(rTNT_i - \mu_{rTNT_j})^2}{2\sigma_{rTNT_j}}$$

where μ_{rTNT_j} and $\sigma^2_{rTNT_j}$ are the means and variances of the two Gaussian distributions estimated from $rTNT$ measures computed on data recorded during the calibration phase.

3.1.2. Mapping data features onto model observations. After each flash, the machine receives 2 values at a time (log likelihood of Riemannian distance between target and non target). To transform such data into a discrete input that is fed to Active Inference, i.e., the set of observations $\mathbb{O}$ with high and low confidence (see reminder in figure 3), we do the following. On training data, we first calculate the log-likelihood ratio or a difference $(lf_0 - lf_1)_i$ per class (target, non-target) at each trial or flash i, and from it we calculate a threshold ρ_T for target and ρ_{NT} for non-target. To compute thresholds (using the same training data as for calculating Riemannian distance), we use the Median Absolute Deviation (MAD). MAD is a more robust estimator of scale than the sample variance or standard deviation, and it works better with non normal distributions. Let us denote Md the median of the distribution and L_i a random event or $(lf_0 - lf_1)_i$ drawn at each trial i, then MAD is referred to as $\rho(L) = Md(|L_i - Md(L)|)$. For pairs $(lf_0, lf_1)_i$ that correspond to target, we denote $\rho(L)_T = \rho_T$ and separately, we calculate MAD for non-target observations, and denote it $\rho(L)_{NT} = \rho_{NT}$. However, if the training set does not possess a sufficient number of samples, outliers will have a strong impact on these estimations. This means that the distribution of the classifier output might differ significantly from the test set, and it could be hard to

generalize the resulting observations. Therefore, in order to get a more robust MAD estimate, we approximate the distributions of $(lf_0 - lf_1)_T$ and $(lf_0 - lf_1)_{NT}$ with beta distributions, using a maximum likelihood estimate. This yields the mean and variance parameters for both distributions. Thanks to this approach we obtain a more robust calibration and less variability between participants. Note that for each subject, we calculate these individual thresholds from their training dataset.

From the testing data, at each trial $j = 1..M$, we draw with repetition a random likelihood pair $(lf_0 - lf_1)_j$ or L_j. Then depending on how it compared with the pre-determined thresholds (ρ_T and ρ_{NT}), we assign an observation category $\phi(L_j) \in \mathbb{O}$, as follows:

$$\phi(L_j) = \phi(lf_0 - lf_1)_j =$$

$$\mapsto \begin{cases} o^{(6)}, \textit{target} \text{ 'T1', if } L_j \geq \rho_T \\ o^{(10)}, \textit{non-target} \text{ 'NT1', if } L_j \leq \rho_{NT} \\ o^{(9)}, \text{ 'NT0', if } L_j \in (\rho_{NT},\ \rho_{NT} + \frac{1}{4}\Delta\rho] \\ o^{(7)}, \text{ 'T0', if } L_j \in [\rho_T - \frac{1}{4}\Delta\rho,\ \rho_T) \\ o^{(8)}, \text{ TXX, if } L_j \in (\rho_{NT} + \frac{1}{4}\Delta\rho,\ \rho_T - \frac{1}{4}\Delta\rho) \end{cases} \tag{5}$$

where $\Delta\rho = \rho_T - \rho_{NT}$.

If $L_j \geq \rho_T$ then L_j represents a target with high confidence 'T1', also if $L_j \leq \rho_{NT}$, then L_j represents a non-target with high confidence 'NT1'. The undefined 'TXX' are placed half way between the two thresholds ρ_{NT} and ρ_T, while we equally divided this distance for less confident observations. 'NT0' and 'T0' are respectively half-way between ρ_{NT} and ρ_T, see Fig. 6. Such observations (with degrees of confidence) are then fed to Active Inference framework.

The raw EEG data that correspond to responses to flashing letters are first classified into target or non-target responses using Riemannian distance estimates between covariance matrices, and a Naive Bayes classifier. Note that in the current model the more or less "noisy" nature of the EEG data is accounted for through considering a confidence level associated with the classification (high, low, undefined).

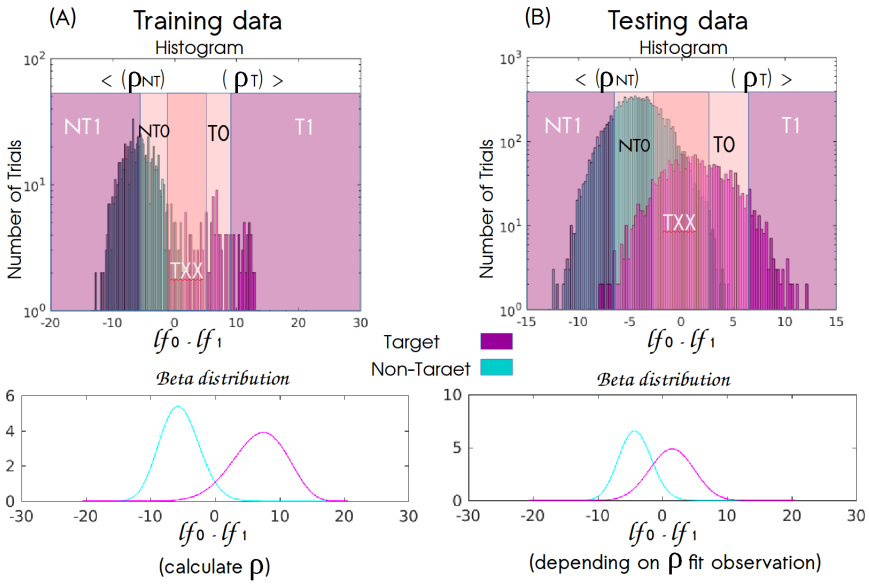

Figure 6: Data from subject 13, likelihood distributions for each trial or flash stimulus $(lf_0 - lf_1)_i$ for training set (A), and $(lf_0 - lf_1)_j$ for testing set (B). On training data, we calculate a threshold using Median Absolute Deviation (MAD) per class, denoted as ρ_T for target and ρ_{NT} for non-target class. Then, with such thresholds learned on training, we map confidence of observations from testing data. Bellow: the beta distributions for a more precise calculation of thresholds.

3.1.3. Simulations We simulate the spelling of 1200 random letters per subject. For each target, Active Inference runs until it decides to spell a letter (without ErrP classifier) or runs until it finds a correctly spelled letter (with Errp classifier). If it flashes the row or column which contains the target, we randomly fetch a target pair $(lf_0 - lf_1)_T$ from our testing dataset. Similarly, if it flashes a column or row that does not contain the target, we will fetch a random non target pair $(lf_0 - lf_1)_{NT}$ from our test dataset. We then map it with ϕ onto one of our z=*6..10* (target/non-target) observations from set $\mathbb{O}$. After this mapping, the pair may turn out to fall in the wrong class depending on the quality of the likelihood pair. As we are picking data randomly, after a consecutive flash, we cannot choose to pick a refracted P300 from our data, and provide more realistic scenario. Hence, we are obliged to set a limit to Active Inference choice of flashing by preventing it from flashing a row/column consecutively.

Note that for simulating an ErrP classifier, the possibilities of describing feedback (ErrP) data with a beta distribution are very large including many possible combinations. Hence, we simply create probabilities to choose a correct or incorrect letter truly or falsely with different specificity and sensitivity levels, as reminder see 2.4.3.

Prior to the testing phase, we assigned the following values to the model parameters:

(i) Calculating likelihood matrix A :
Matrix **A** expresses the probability of each observation category, given each possible state value. It is computed individually, from the training data of each subject. In our simulations,

we draw N_T=*2000* samples of target data, N_{NT}=*2000* samples of non target data. We then computed the proportion of samples who fell into each observation category in order to set the above probabilities.

(ii) Setting values for C

Differently from matrix A, values chosen for C are same for all subjects. We assign a high value to a correctly spelled letter, 'FC1', and penalize the wrongly spelled 'FI1', (for a reminder, see Fig. 4). Here we discuss the empirical evaluation of the distance between the extreme values assigned to observations, i.e., penalty and preferences. For instance, how strong should be the penalty for incorrect feedback 'FI1', 'FI0' and 'FXX'. Observations (target or not) are valued equally (zero vector) $o_T(i) = 0$, where *i=6,..10* (as reminder of observations, see figure 3). In contrast, we vary values for feedback observations (correct or not), as follows. A quadratic function $g(d) = d^2$, $d \in [1,2,..5]$ maps the penalty to the observations, and a parameter κ, regulates such penalty: $o_T(j) = \kappa + g(d)$, for *j=1,..5*. For instance, the strongest penalty is when $g(d) = 5^2$ is set for an incorrect feedback with high confidence (FI1); κ is a parameter influencing the penalty that we vary for 3 distinct subjects, see Figure 7.

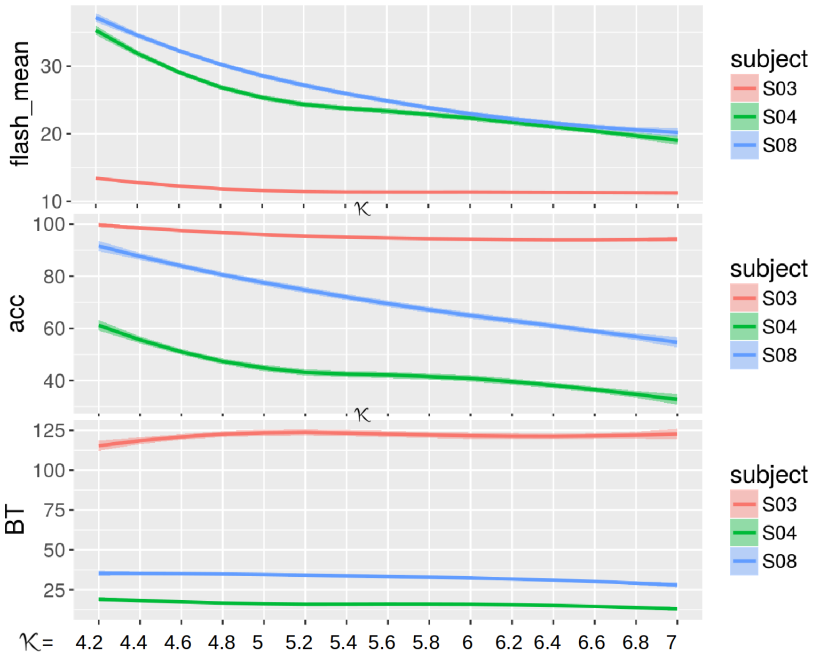

Figure 7: Varying κ in C vector to demonstrate difference in speed (flash_mean), in accuracy (acc), and Bit Rate (BT) for 3 subjects (S03 - good, S08 - bellow average, S04 - poor classification performance).

By augmenting κ, we can decrease the distance between the feedback correct(max) and incorrect(min). Note that the smaller the distance (higher κ), the faster the spelling with less accuracy which in total does not significantly affect the bit rate (BT). For all subjects, we empirically fixed $\kappa = 5.6$ i.e. between $\kappa \in [4,7]$, a range of values that we

determined empirically and for which Active Inference is stable and exhibits the expected type of behaviour.

(iii) Selecting values for precision, γ

For our thorough evaluation, we considered a unique prior distribution over the precision parameter γ, for all subjects, with $\alpha = 1$ and $\beta = 128$. To illustrate the effect of this parameter though, we performed a few simulations with three different subjects (S03, S04, S08), varying its prior distribution. For $\alpha \in (1,..,128)$ and $\beta \in (1,..,128)$, we performed all the combinations and did not observe any significant change in accuracy, flash mean nor bit-rate, see Figure 8. This is because of our choice of transition values being either 0 or 1 (high confidence) in the **B** matrix, i.e., the γ parameter in that case has very little influence on the choice of future action and hidden state.

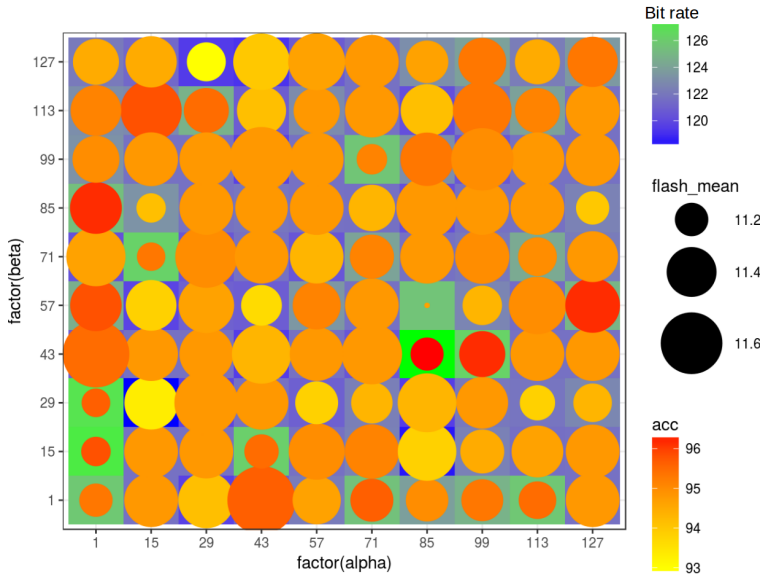

Figure 8: Varying α and β parameters of γ precision. The rectangle colour denotes bit-rate, the size of circles denotes flash mean, and the circle colours denote the accuracy.

3.1.4. Evaluation Metrics We test the following Active Inference (AI) models:

- basic AI (optimal stopping and flashing);
- basic AI + *lookAway*.

To examine the performance rates of basic AI +*lookAway*, in our simulation (same, for 12000 "letters"), instead of selecting random letters as target, we set *lookAway* as the only target "letter" (12000 "*lookAways*").

- basic AI + realistic ErrP classifier;
- basic AI + perfect ErrP classifier.

The ErrP classifier output contains purely simulated data (both perfect and more realistic).

We compare these AI models with a fairly standard approach based on naïve Bayes classifier with two variants for actions:

1. No adaptive actions, i.e. a fixed number of flashes (12 repetitions fixed *a priori*) and pseudo-random flashing, denoted as *fixed-flash*;

2. An optimal strategy based on a threshold on the maximum a posteriori (MAP) with pseudo-random flashing, denoted as *optimal stopping* as in [21]. As mentioned in the related works, optimal stopping spells a letter once the accumulated evidence about a letter reaches a predefined confidence threshold or certainty. We implemented different threshold values (between $0.8, 0.9$, 0.95 and 0.99). For comparison, we chose 0.9 as it yielded the highest bit rate on average in our dataset.

Note that all approaches apply on the same features – Riemannian distance of covariance matrices, as described above in the subsection *3.1.1 Features*.

———————-

Simulations were performed on data collected from the simulated spelling of 12000 letters with 18 subjects, who were recorded in a previous experiment [17]. Even though during the present simulations it took milliseconds to fetch features from the hard drive, to compare the performance of the various algorithms we considered the speed of the flash during the original experiment, i.e. 0.2s, and we measured the bit rate accordingly for each subject. The amount of bits (b) transferred is given by:

$$b = log_2(K) + P \cdot log_2(p) + (1-p) \cdot log_2(\frac{1-P}{N-1})$$

with K: number of possible choices (classes) and p: P300 classifier accuracy. Considering that each flash lasts 0.2s, the time T it takes to spell a letter is hence $0.2 \times Nb_{flash}$, thus the bit rate br indicates the BCI information transfer rate in bit/min with: $br = b \times \frac{60}{T}$ – see [34].

We tested to which extent the performance (as measured by bit rate) of optimal flashing outperforms classical P300 algorithms. We performed a one-way analysis of variance (ANOVA) with repeated measures and post-hoc Tukey with false discovery rate correction [35] enabling a clear differentiation between algorithms. Independent variable: algorithm (6 groups: 4 Active Inference + 2 standard), dependent variable: bit rate. The threshold of significance is set at $p < 0.01$.

4. Results

We present the comparison of AI instances with standard P300-speller algorithms using their average bit rate values, see table 1, and see figure 9.

Methods	Fixed-flash	Optimal Stop	AI basic	AI ErrP perfect	AI ErrP real	AI *lookAway*
Bit rates	10.49 b/m	45.86 b/m	54.32 b/m	73 b/m	64.94 b/m	51.85 b/m

Table 1: Table with bit-rate average values of all methods.

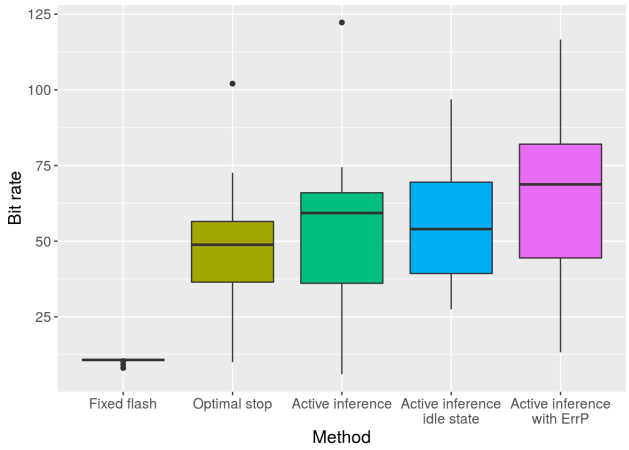

Figure 9: Comparison in bit rate (bit/min) between fixed flash, optimal stopping 0.9, AI basic, AI of lookAway, and AI + realistic ErrP. All methods significantly differ from one another $(p < 0.01)$.

Active inference has lower accuracy rates on average than Optimal Stop (71.97% vs 76.62%, Figure 10.B.), however it is a lot faster (18.51 vs 24.88 flashes, Figure 10.C.).

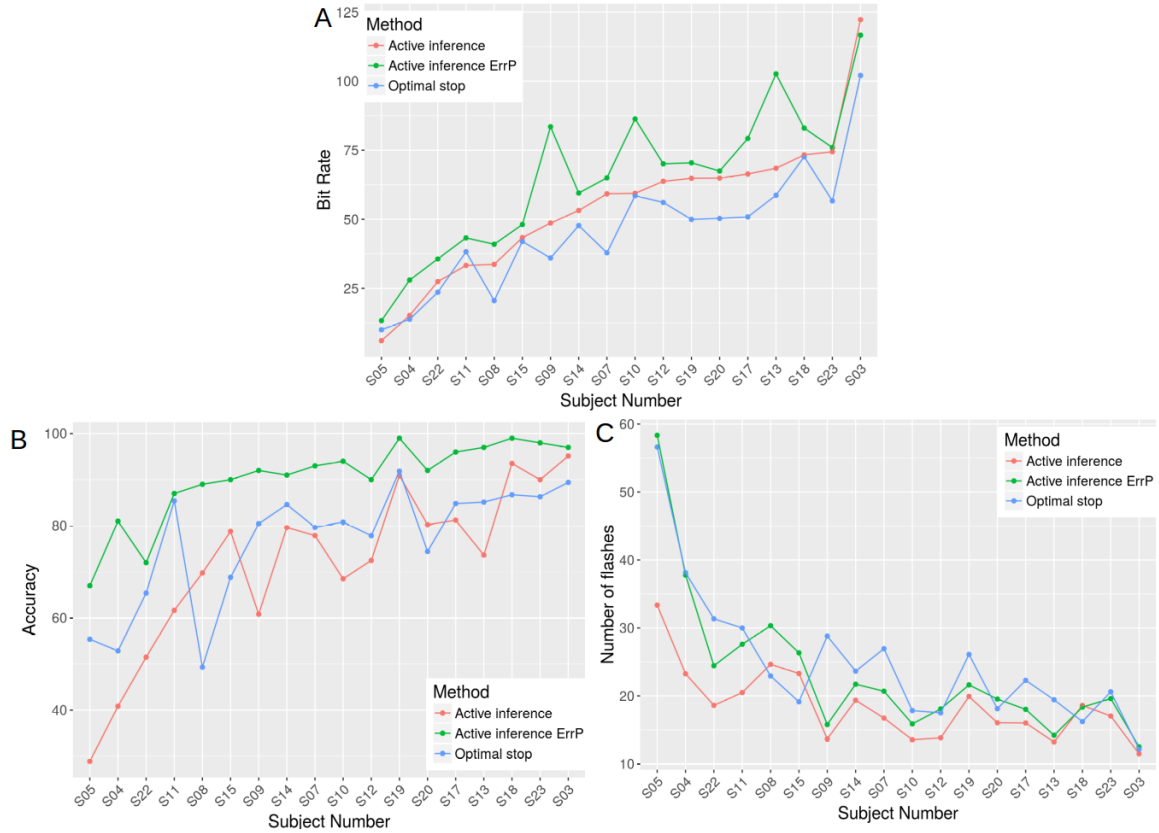

Figure 10: Comparison of (A.) Bit Rate, (B.) Accuracy and (C.) Mean number flashes, of Active inference basic (in red), AI with realistic ErrP classifier (in green) and Optimal stopping (in blue) across subjects (sorted by bit rate from left to right).

It is interesting to see how Active Inference adapts its flashing pattern depending on the certainty of the observations. In Figure 11 we compared side by side subjects with poor (S04) and good (S03) classification performance.

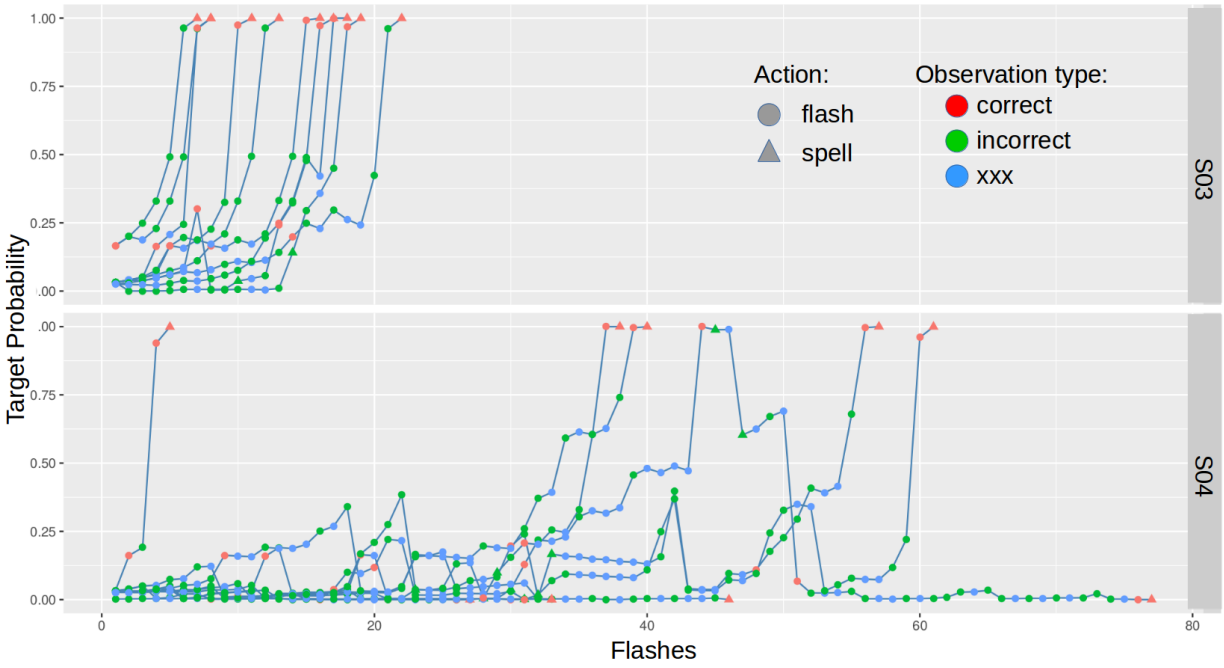

Figure 11: Progression of probabilities of letters during flashing of AI with realistic ErrP. Top: a subject with good performance, S03, bottom: poor performance one, S04. Each curve corresponds to one letter and ends with a red triangle that represents a "correctly" spelled letter (it can be wrong). If the curve ended with a triangle while in low probability it represents a wrong assumption. The red refers to both correct feedback (red triangle) and target (red circle), while green (incorrect) refers to both incorrect feedback (green triangle) and non-target observations (green circle). The undefined target/non-target is in blue. The frequency of error is evident with subject S04 while there are no error present with S03.

When studying Active inference with ErrP, we noticed that at least a 75% accurate ErrP classifier (with specificity = sensitivity) is necessary for Active Inference to outperform other algorithms (see Figure 12).

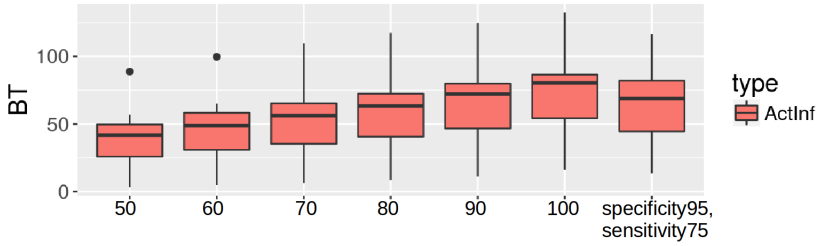

Figure 12: Bit Rate increase with feedback classifier's accuracy – from 50 to 100 %

5. Discussion

The naive algorithm (Fixed flash) achieves only (10.49b/m). Active Inference showed a significantly higher bit rate $(54.32 bit/min)$ than optimal stopping $(45.86 b/m)$, giving an increase of about 17%. Active Inference performance increased even further when comparing to optimal stopping by 58% when a perfect ErrP classifier with 100% accuracy is used $(73 b/m)$. However, this perfect classifier being over optimistic, we considered a more realistic one with specificity 0.95 and sensitivity 0.75 $(64.94 b/m)$; resulting with an increase of about 41% when comparing to optimal stopping. When only idle user or "lookAway" states are simulated, it accurately "switches off" the speller about 90% of the time, after about 24 flashes $(51.85 bit/min)$. A natural consequence is that the *LookAway* state often requires more flashes than any other user intention to be inferred. That is because Active Inference observes the 37^{th} state, but as it in fact does not exist, it does not elicit a real observation (there is no 37^{th} letter), it will keep receiving non-target responses when flashing. When looking at the performance (bit rate) of Active Inference subject per subject it behaves worse than Optimal Stopping for 2 out of 18 of them (i.e. S05 and S11). Interestingly, those are among the subjects with lowest bit rate, see Figure 10.A. It can be explained by the fact that Active Inference has a short observation time (see Figure 10.C.), if it receives a consecutive number of observations with low probability (e.g. undefined observation TXX). However, the speed-accuracy trade-off can be regulated within the vector C by setting a stronger penalty to wrongly spelled letters. We also observe that the noisier the EEG data, the lower the performance, whatever the methods.

5.1. Perspectives

Due to the lack of ground truth, at the moment we simulated only two observations for the ErrP classifier: correct or incorrect feedback (in/correctly spelled letter) with high confidence, "FC**1**" and "FI**1**" with a degree of specificity and sensitivity (75% and 95%), but not low confidence "FC**0**" or "FI**0**". In a more realistic scenario, Active Inference would benefit from an increased variety of feedback observations, i.e. correct / incorrect feedback with low confidence and "undefined". In this case its distribution would be calibrated during training, as we did with target and non-target observations.

Clearly, the fact that this is a simulation is a drawback, as we have no way of controlling the refractory effect for instance. We account for this phenomenon in our simulations by forbidding two consecutive stimuli (which effects in a slight reduction in performance of Active Inference). In the future, we would account for an additional observation representing the decrease in the P300 amplitude with repetition or frequency.

Another constraint with Active Inference is that we must tune all the mentioned parameters as priors beforehand. We presented an application where only the likelihood is learned for each subject while other variables were empirically selected and kept the same for all subjects. In order to learn a sensible range for those parameters and validate our model, we first tested Active Inference on purely simulated data, in [36]. Note that the current instantiation of the Active Inference we presented is not fully adaptive yet. Precisely, it is

adaptive as actions are driven online by ongoing observations, however it is not endowed with (long term) learning in the sense that, for now, only model states are updated, not model parameters.

Active Inference shows promising results in terms of performance, proposing a generic framework in order to build a flexible machine that articulates inference (perception) and decisions (actions) to optimize the interaction with the user. Yet, while the resulting system may trigger (positive) adaptive behaviour from the user, currently it does not account explicitly for user's evolution over time, nor does it anticipate it. Future work will consist in creating such "co-adaptive" system. For instance [37, 38] provide theoretical models that set evidence of mutual, co-adaptive learning of the machine and user. [38] highlights that the machine should not learn too fast as it would provide sub-optimal performance. Such co-adaptation is indeed possible with our framework, for instance by adding new hidden states, e.g. user fatigue or learning, as well as by updating model parameters online, such as the likelihood matrix A.

We could envision to use an additional "layer" of active inference to implement a language model for word auto-completion in a P300-speller. In such case the set of hidden states could be increased with another one, referring to a correctly spelled word, along with the correctly spelled letter. And, the desired outcome (in vector C) would correspond to a "correctly spelled word" instead of or along with the correctly spelled letter.

Future developments would consist in testing Active Inference online, also designing, testing, as well as applying Active Inference to other BCI paradigms, such as a Motor Imagery BCI.

6. Conclusion

In this paper, we propose the use of Active Inference, a generic Bayesian framework used as a computational model of brain processes. If endowed to the machine, Active Inference has the potential to be applied on various BCI tasks, to adapt the machine to the user not only by adjusting to signal variability (adapting the signal processing pipeline) but by modeling and acting upon its causes (here a simplified example of user states, that are, user intentions in a P300 context). We show that it is very flexible and generic, and demonstrate it *via* a P300 speller simulation on real-data. Furthermore, we demonstrate superiority of Active Inference when compared with well known P300-speller approaches.

To make use of Active Inference one must specify: what the machine observes, here, classified P300 or Error Potential features for instance; what the machine infers, here, the user intentions to spell or pause; and what the machine performs as action, here, to flash, spell or switch-off the application for example, and finally what is the overreaching goal, here, to spell a correct letter with high confidence. With such information provided to Active Inference, it builds confidence through observations, predicts user intentions, and chooses the optimal action to minimize prediction error and reach a desired outcome or goal. As consequence of applying Active Inference in a P300-speller context, it performs optimal flashing and stopping, that is, automatic flashing of such letters that maximize information (minimize entropy) about

the target letter, and stopping once the goal (correctly spelled letter) is reached. We support our choice for adding yet another method for adapting a BCI as it offers a vast range of adaptation possibilities and flexibility, while minimizing only one objective function, the *free energy*.

Although we illustrated the potential of Active Inference for BCI in the special case of P300-based BCI, it is important to emphasize the genericity of this approach. Essentially, our current implementation, relates EEG data classification outputs to inferred hidden states that will trigger certain state transitions through actions. This is a very generic process where EEG data features could be of any kind, such as ERPs but also frequency specific induced activities like in motor-imagery based BCIs.

The overarching goal is to "influence" the user through optimal machine action in order to fulfill efficiently user's intent. We envision that Active Inference could unify most approaches and paradigms in one adaptive BCI framework, as conceptualized in [25].

Appendix

Appendix 1:

Relative entropy, also called the Kullback-Libeler divergence, D_{KL} of 2 probability density functions Q and P : $D_{KL}(Q\|P)$ is a measure of the information gained when one revises one's beliefs from the prior probability distribution P to the posterior probability distribution Q. In other words, it is the amount of information lost when Q is used to approximate P [39]. In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated distribution, in our case being $P(s_i|o_i,m)$, given the model m. Q typically represents an approximation of P, or in our case $Q(s_i|m)$. In order to find a distribution Q that is "closest" to P, we can minimize the KL divergence and compute an information projection $p^* = \arg\min_{p\in P} D_{KL}(q\|p)$. Viewing the KL divergence as a measure of distance in the space of probability distributions, p^* is the "closest" distribution to q of all the distributions in P. However, note that the KL divergence is not a metric as it is non-symmetric, in general $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$, and does not satisfy the triangle inequality. The KL divergence is always non-negative $D_{KL} \geq 0$, and is equal to zero if and only if the two distributions are equal. For discrete probability distributions Q (posterior) and P (prior), KL divergence is defined to be [40]:

$$D_{KL}(Q\|P) = -\sum_i Q_i \, log \, \frac{P_i}{Q_i}$$

Appendix 2:

As the agent is a Bayesian modeler, at each step it wants to maximize the model evidence or minimize surprise, i.e., to minimize prediction error $E_{Q(o_\tau|\pi)}[D_{KL}[Q(s_\tau|o_\tau,\pi)|Q(s_\tau|\pi)]]$. To evaluate surprise is a difficult problem of **exact** Bayesian inference, because we need to minimize the prediction between potentially many future states again given many possible priors. One needs to find a bound for the marginal (i.e., integrated) likelihood, which generally involves an intractable integral over hidden states s_i, i.e., summing out the states from $Q(s_i,o_i)$.

So we need approximations or a bound to solve it (for more information, see [41]). Thus, if we add the same fixed, variational approximate distribution Q in the surprise, we get an approximate solution to the marginal likelihood and we get the expectation of surprise within a bound. Such minimization of surprise is also called the variational (approximate) free energy.

ACKNOWLEDGMENT

This work was financially supported by BCI-Lift Inria project, and the French National Research Agency (ANR-11-LABX-0042, ANR-11-IDEX-007 and ANR-18-CE28-0016). The data used in this study were acquired as part of the French ANR project ANR-DEFIS 09-EMER-002 CoAdapt.

References

[1] Jose del R Milan and Jose M Carmena. Invasive or noninvasive: Understanding brain-machine interface technology [conversations in bme]. *IEEE Engineering in Medicine and Biology Magazine*, 29(1):16–22, 2010.

[2] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.

[3] Thorsten O Zander and Christian Kothe. Towards passive brain–computer interfaces: applying brain–computer interface technology to human–machine systems in general. *Journal of neural engineering*, 8(2):025005, 2011.

[4] J-M Batail, S Bioulac, F Cabestaing, C Daudet, D Drapier, M Fouillen, T Fovet, A Hakoun, R Jardri, C Jeunet, et al. Eeg neurofeedback research: A fertile ground for psychiatry? *L'Encéphale*, 2019.

[5] Eric W Sellers, Dean J Krusienski, Dennis J McFarland, Theresa M Vaughan, and Jonathan R Wolpaw. A p300 event-related potential brain–computer interface (bci): the effects of matrix size and inter stimulus interval on performance. *Biological psychology*, 73(3):242–252, 2006.

[6] Benjamin Blankertz, K-R Muller, Gabriel Curio, Theresa M Vaughan, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, et al. The bci competition 2003: progress and perspectives in detection and discrimination of eeg single trials. *IEEE transactions on biomedical engineering*, 51(6):1044–1051, 2004.

[7] Haline E Schendan, Nancy G Kanwisher, and Marta Kutas. Early brain potentials link repetition blindness, priming and novelty detection. *Neuroreport*, 8(8):1943–1948, 1997.

[8] Jing Jin, Eric W Sellers, and Xingyu Wang. Targeting an efficient target-to-target interval for p300 speller brain–computer interfaces. *Medical & biological engineering & computing*, 50(3):289–296, 2012.

[9] Caterina Cinel, Riccardo Poli, and Luca Citi. Possible sources of perceptual errors in p300-based speller paradigm. *Proceedings of the 2nd International BCI Workshop and Training Course*, 2004.

[10] Jun Qu, Fei Wang, Zhenping Xia, Tianyou Yu, Jing Xiao, Zhuliang Yu, Zhenghui Gu, and Yuanqing Li. A novel three-dimensional p300 speller based on stereo visual stimuli. *IEEE Transactions on Human-Machine Systems*, 2018.

[11] Tobias Kaufmann, SM Schulz, Claudia Grünzinger, and Andrea Kübler. Flashing characters with famous faces improves erp-based brain–computer interface performance. *Journal of neural engineering*, 8(5):056016, 2011.

[12] Thibault Verhoeven, Pieter Buteneers, JR Wiersema, Joni Dambre, and PJ Kindermans. Towards a symbiotic brain–computer interface: exploring the application–decoder interaction. *Journal of neural engineering*, 12(6):066027, 2015.

[13] BO Mainsah, G Reeves, LM Collins, and CS Throckmorton. Optimizing the stimulus presentation paradigm design for the p300-based brain-computer interface using performance prediction. *Journal of neural engineering*, 14(4):046025, 2017.

[14] Mathew Salvaris and Francisco Sepulveda. Visual modifications on the p300 speller bci paradigm. *Journal of neural engineering*, 6(4):046011, 2009.

[15] SC Kleih, F Nijboer, S Halder, and A Kübler. Motivation modulates the p300 amplitude during brain–computer interface use. *Clinical Neurophysiology*, 121(7):1023–1031, 2010.

[16] Hannes Verschore, Pieter-Jan Kindermans, David Verstraeten, and Benjamin Schrauwen. Dynamic stopping improves the speed and accuracy of a p300 speller. In *International Conference on Artificial Neural Networks*, pages 661–668. Springer, 2012.

[17] Jérémie Mattout, Margaux Perrin, Olivier Bertrand, and Emmanuel Maby. Improving bci performance through co-adaptation: applications to the p300-speller. *Annals of physical and rehabilitation medicine*, 58(1):23–28, 2015.

[18] Hendrik Woehrle, Mario M Krell, Sirko Straube, Su Kyoung Kim, Elsa A Kirchner, and Frank Kirchner. An adaptive spatial filter for user-independent single trial detection of event-related potentials. *IEEE Transactions on Biomedical Engineering*, 62(7):1696–1705, 2015.

[19] Pieter-Jan Kindermans, Hannes Verschore, David Verstraeten, and Benjamin Schrauwen. A p300 bci for the masses: Prior information enables instant unsupervised spelling. In *Advances in Neural Information Processing Systems*, pages 710–718, 2012.

[20] Pieter-Jan Kindermans, Michael Tangermann, Klaus-Robert Müller, and Benjamin Schrauwen. Integrating dynamic stopping, transfer learning and language models in an adaptive zero-training erp speller. *Journal of neural engineering*, 11(3):035005, 2014.

[21] Perrin Margaux, Maby Emmanuel, Daligault Sébastien, Bertrand Olivier, and Mattout Jérémie. Objective and subjective evaluation of online error correction during p300-based spelling. *Advances in Human-Computer Interaction*, 2012:4, 2012.

[22] Aniana Cruz, Gabriel Pires, and Urbano J Nunes. Double errp detection for automatic error correction in an erp-based bci speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(1):26–36, 2018.

[23] Dmitry Kalika, Leslie M Collins, Chandra S Throckmorton, and Boyla O Mainsah. Adaptive stimulus selection in erp-based brain-computer interfaces by maximizing expected discrimination gain. In *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pages 1405–1410. IEEE, 2017.

[24] Boyla O Mainsah, Kenneth D Morton, Leslie M Collins, Eric W Sellers, and Chandra S Throckmorton. Moving away from error-related potentials to achieve spelling correction in p300 spellers. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(5):737–743, 2015.

[25] Jelena Mladenović, Jérémie Mattout, and Fabien Lotte. A generic framework for adaptive eeg-based bci training and operation, 2017.

[26] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127, 2010.

[27] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.

[28] Thomas HB FitzGerald, Philipp Schwartenbeck, Michael Moutoussis, Raymond J Dolan, and Karl Friston. Active inference, evidence accumulation, and the urn task. *Neural computation*, 27(2):306–328, 2015.

[29] Karl Friston, Philipp Schwartenbeck, Thomas FitzGerald, Michael Moutoussis, Timothy Behrens, and Raymond J Dolan. The anatomy of choice: dopamine and decision-making. *Phil. Trans. R. Soc. B*, 369(1655):20130481, 2014.

[30] Bertrand Rivet, Antoine Souloumiac, Virginie Attina, and Guillaume Gibert. xdawn algorithm to enhance evoked potentials: application to brain–computer interface. *IEEE Transactions on Biomedical Engineering*, 56(8):2035–2043, 2009.

[31] J Mattout, G Gibert, V Attina, E Maby, and O Bertrand. Probabilistic classification models for brain computer interfaces. In *Proceedings of the Human Brain Mapping Conference, Melbourne, Australia*, volume 1519, 2008.

[32] Alexandre Barachant and Marco Congedo. A plug&play p300 bci using information geometry. *arXiv preprint arXiv:1409.0107*, 2014.

[33] Marco Congedo, Alexandre Barachant, and Rajendra Bhatia. Riemannian geometry for eeg-based brain-computer interfaces; a primer and a review. *Brain-Computer Interfaces*, 4(3):155–174, 2017.

[34] Peng Yuan, Xiaorong Gao, Brendan Allison, Yijun Wang, Guangyu Bin, and Shangkai Gao. A study of the existing problems of estimating the information transfer rate in online brain–computer interfaces. *Journal of neural engineering*, 10(2):026014, 2013.

[35] William S Noble. How does multiple testing correction work? *Nature biotechnology*, 27(12):1135–1137, 2009.

[36] Jelena Mladenović, Mateus Joffily, Jérémy Frey, Fabien Lotte, and Jérémie Mattout. Endowing the machine with active inference: A generic framework to implement adaptive bci. In *NeuroAdaptive Technology Conference'17*, 2017.

[37] Josh Merel, Donald M Pianto, John P Cunningham, and Liam Paninski. Encoder-decoder optimization for brain-computer interfaces. *PLoS computational biology*, 11(6):e1004288, 2015.

[38] Jan Saputra Müller, Carmen Vidaurre, Martijn Schreuder, Frank C Meinecke, Paul Von Bünau, and Klaus-Robert Müller. A mathematical model for the two-learners problem. *Journal of neural engineering*, 14(3):036005, 2017.

[39] KENNETH P Burnham and DAVfD R Anderson. A practical information-theoretic approach. *Model selection and multimodel inference, 2nd ed. Springer, New York*, 2002.

[40] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[41] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.