Subject-Independent ERP-Based Brain–Computer Interfaces

Kha Vo, Thuy Pham, Diep N. Nguyen, Ha Hoang Kha, and Eryk Dutkiewicz

***Abstract*—Brain–computer interfaces (BCIs) are desirable for people to express their thoughts, especially those with profound disabilities in communication. The classification of brain patterns for each different subject requires an extensively time-consuming learning stage specific to that person, in order to reach satisfactory accuracy performance. The training session could also be infeasible for disabled patients as they may not fully understand the training instructions. In this paper, we propose a unified classification scheme based on ensemble classifier, dynamic stopping, and adaptive learning. We apply this scheme on the P300-based BCI, with the subject-independent manner, where no learning session is required for new experimental users. According to our theoretical analysis and empirical results, the harmonized integration of these three methods can significantly boost up the average accuracy from 75.00% to 91.26%, while at the same time reduce the average spelling time from 12.62 to 6.78 iterations, approximately to two-fold faster. The experiments were conducted on a large public dataset which had been used in other related studies. Direct comparisons between our work with the others' are also reported in details.**

***Index Terms*—Event-related potentials, EEG, P300-Speller, brain-computer interface, dynamic stopping, adaptive learning.**

I. Introduction

FARWELL [1] first introduced a brain-computer interface (BCI) called P300-Speller (P3S) to help people elicit spelling letters only by their mere thoughts. A P3S exploits the characteristics of visual P300 responses [2], [3], which are one of the most dominant types of event-related potentials (ERP). Since then, numerous developments had been carried out based on [1] with the main targets of increasing the accuracy, or reducing the experimental time. One approach is to optimize the classification algorithm [4], [5]. Other techniques focus on extracting the relevant features of EEG signals [6], [7], to modify the visual flashing paradigm [8], [9], to deploy an online learning model [10]–[12] or to add language model plug-ins [13] into the existing framework.

Manuscript received June 6, 2017; revised January 2, 2018 and February 13, 2018; accepted February 13, 2018. Date of publication February 28, 2018; date of current version April 6, 2018. *(Corresponding author: Kha Vo.)*

K. Vo, T. Pham, D. N. Nguyen, and E. Dutkiewicz are with the Global Big Data Technologies Centre, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: anhkha.vo@student.uts.edu.au; diep.nguyen@uts.edu.au; thuy.pham@uts.edu.au; e.dutkiewicz@uts.edu.au).

H. H. Kha is with the Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City 6103, Vietnam (e-mail: hhkha@hcmut.edu.vn).

Digital Object Identifier 10.1109/TNSRE.2018.2810332

There had been studies of P3S on disabled subjects. Hoffmann [14] implemented a six-choice P3S on 5 disabled users, using Bayesian Linear Discriminant Analysis (BLDA) as their main classification method. Sellers [15] performed a simple four-choice P3S experiment on three amyotrophic lateral sclerosis (ALS) patients. Mainsah *et al.* [16] integrated a dynamic stopping scheme into P3S and conducted the experiments on 10 ALS subjects. Clements *et al.* [17] analyzed the effects of wet and dry sensors on 8 subjects with communication difficulties. Although different in their frameworks, all of those mentioned works shared a common result that the accuracy performing on ALS or disabled subjects was much lower than those of healthy ones. The reason is that P300 responses of disabled or ALS patients are not as discriminative as those of healthy subjects [14]–[16]. Another drawback of those mentioned works is that each new user needs to perform an extensive training stage for the system to recognize their individual brain patterns, before actually testing it.

As a result, we are motivated by the idea that subject-independent BCI systems are desirable, for both healthy and disabled subjects. More specifically, new users to the system are not required to perform the exhaustive training stage. This would save a huge amount of time and create easy usage for new subjects, hence make ERP-based BCIs more practical in real-life everyday use. Indeed there had been a few notable approaches to fulfill this idea, proposed by Pieter-Jan *et al.* [12] and Kindermans *et al.* [18], which used no training samples to form the classifier. The weakness of this method is that it performs poorly in terms of accuracy, as compared to the methods with pre-trained classifiers. It is also unnecessary to use no prior data to construct the classifier, since the EEG datasets are abundantly available. The right question for being solved in our case is that how can we adapt the existing model with the new subject-specific samples, rather than the question of how to form a classifier from scratch during an on-line experiment.

The main contribution of this paper are as follows.

- Propose a novel unified method of ensemble classifier (EN), dynamic stopping (DS), and adaptive learning (AL). DS validates the classifier's confidence about its decision to stop the flashing paradigm at any moment. The newly-classified trials obtained by DS are exploited by AL to adapt the existing classifier to the current subject. Because

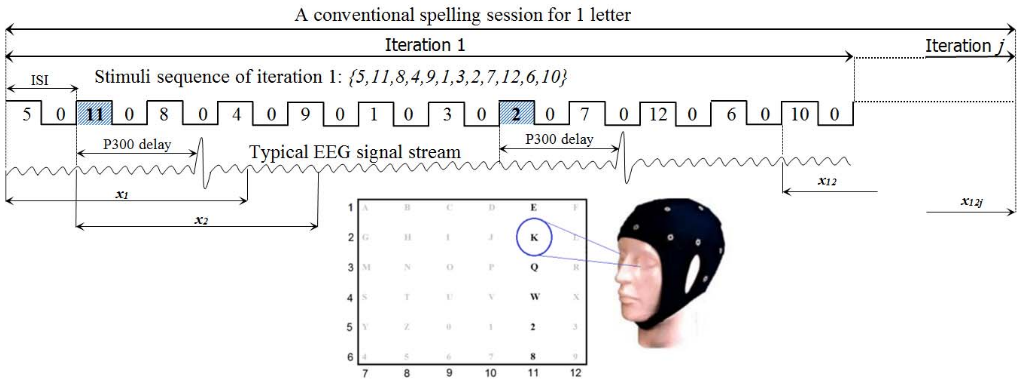

Fig. 1. The EEG signal timeline and row-column flashing paradigm. The subject is focusing on the letter 'K' of the P3S, hence the target row is row 2 and the target column is column 11 of the P300 matrix. The main signal stream is depicted by sequential flashes of numbered rows and columns (from 1 to 12 in shuffled order), interleaved with unflashed intervals (numbered 0). In any iteration, the post-stimulus feature vectors associated to row 2 and column 11 contain the P300 responses. Specifically, the target flashes are shaded blue in the stream of Iteration 1, and their corresponding feature vectors are denoted by $\boldsymbol{x}_8$ and $\boldsymbol{x}_2$. These are ERPs occurs approximately 300 ms (P300 delay) after the target stimuli onset.

AL requires a huge amount of computational resources during the on-line scenario, EN comes to play as a very important role to diminish this burden. Also, EN allows the classifier to be subject-diverse (in other words, the pre-trained classifier can be composed from multiple subjects).

• Analyze the correlation between theoretical analysis with empirical observations to demonstrate the validity of our proposal.

• Implement for the first time the incremental support vector machine (ISVM) toolbox in the BCI context to get the pre-trained classifiers get adapted to the new experimental subjects.

• Re-conduct the other related methods using the same dataset and experimental setup as in our approach. Therefore a reliable and direct comparison between our work with the related studies is provided.

The rest of this paper is organized as follows: Section II presents the ensemble classification framework implemented on the P3S. Section III proposes two dynamic stopping criteria for the flashing paradigm. Section IV describes the adaptive learning method which is conducted in real-time after the dynamic stopping stage. The dataset descriptions, experimental results, discussions and comparisons with related studies are presented in Section V. Finally, Section VI derives the paper's conclusions.

II. Classification Methodology

This section presents the main framework of our proposed classification scheme. Basically, we have two main stages: the learning (or training) stage to prepare the classifier independent to new subjects (as presented in Section II-C), and the validation (or testing) stage to test their performance (as presented in Section II-D). However we would like first to summarize the general flow of our method in Section II-A and present the dataset partitioning step in Section II-B.

A. Main Classification Framework

An example of the row-column flashing paradigm for the P3S used in our work is presented in Fig. 1. The diagram of

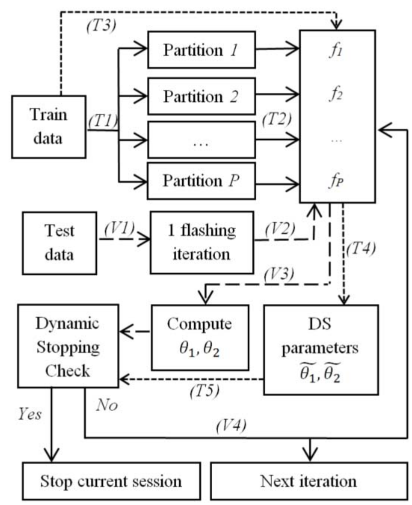

Fig. 2. The main diagram of our proposed classification framework for a one-letter P3S session.

our main framework is shown in Fig. 2. According to Fig. 2, the training (or learning) stage consists of the following steps: (T1) Part of the learning data (as described in more detail in Section V-C and Table I) is divided into P partitions, denoted by T_p (as in Eq.(1)). (T2) The partitioned data is used to construct P ensemble classifiers, as described in Section II-C. (T3, T4) The remaining unused learning data is fed as input for the grid search algorithm (as presented in Algorithm 1) to search for DS parameters $\tilde{\theta}_1$ and $\tilde{\theta}_2$. (T5) The resulted parameters are used as the dynamic stopping criteria (as presented in Section III) for the validation stage.

The validation (or testing) stage consists of the following steps: (V1) The test data is collected through the row-column P3S flashing paradigm after each iteration (which equals to 12 trial vectors). (V2) The preprocessed vectors (as described in Sec V-A) are scored by the ensemble scheme, as in Eq.(5) and Eq.(6). (V3) The resulting scores are used to compute the dynamic stopping variable θ_1 and θ_2, to perform the DS check,

TABLE I

CLASSIFICATION SCHEMES' SETTINGS. P IS THE NUMBER OF TOTAL PARTITIONS (EACH PARTITION HAS LEARNING SAMPLES OF ONLY 1 SUBJECT), L IS THE NUMBER OF LEARNING LETTERS USED IN EACH PARTITION. THE SUBJECTS ARE NAMED AS REPORTED IN [33]. THE NUMBER OF SAMPLES FOR EACH SCHEME IS 2160, EQUALS TO 12 LEARNING LETTERS

Scheme	P	L	Learning Subjects
ENS-2	2	6	ACS, APM
ENS-3	3	4	ACS, APM, ASG
ENS-4	4	3	ACS, APM, ASG, ASR
ENS-6	6	2	ACS, APM, ASG, ASR, CLL, DCM

as in Eq.(8). If the DS criteria are satisfied, the current letter session is stopped and the output is shown. (V4) If the criteria are not satisfied, an adaptive learning stage is implemented (as presented in Section IV), and the next flashing iteration is performed.

B. Dataset Partitioning

Since our purpose is to design a subject-independent system, there is no learning sample required from the experimental subjects. However, our approach is different from the unsupervised method [12], [18]. We employ an off-line pre-trained classifier first, then adaptively update it for the new user, rather than constructing the classifier using only online samples. Therefore, the learning stage using the existing training samples is desired. In this paper, we select support vector machine (SVM) as the baseline learning algorithm, which had been used in various BCI studies [19]–[22]. The reason of choosing SVM, rather than some Bayesian probabilistic approaches [11], [14], [17], [23], [24], is that we want to exploit the computational strength to combine the dynamic stopping check with the adaptive SVM update algorithm [25], [26] into the existing ensemble partitioning framework. This integration can significantly improve the performance as compared to other methods, which will be presented in Section V.

We denote P training sets of P different subjects as

$$T_p = \left\{(\boldsymbol{x}_{p,i}, y_{p,i}) \in \mathbb{R}^D \times \{-1, 1\}\right\}_{i=1}^{N}, \tag{1}$$

where $p \in \{1, \ldots, P\}$, $\boldsymbol{x}_{p,i}$ are the preprocessed training feature vectors of length D, $y_{p,i}$ are the binary labels or classes (-1 for non-P300 responses and 1 for P300 responses), and N is the number of training samples (of both classes) in each training set. Hereinafter we call each training set T_p a *partition*.

C. SVM Classifiers Learning Stage

SVM is a discriminative method dealing with binary classification problems [27]. Given P training partitions denoted by T_p, the classifier solution $(\boldsymbol{\alpha}_p, b_p)$ for partition p is obtained by solving a quadratic convex constrained optimization problem in the duality space [27].

In this paper, we employ the homogeneous polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$, and the quadratic solving program provided by [28]. Since C, the penalty parameter, can be freely chosen and can heavily affect the classification performance, we perform the leave-one-out cross validation process [14] on P classifiers to determine the optimum C_p for each partition to yield the best classification performance on the ensemble training set. The possible values for C_p are 0, 0.01, 0.02, 0.05, 0.1, 0.25, 0.5, 0.75, and 1.

For a new validation feature vector $\boldsymbol{x}$, its *classifier score* generated by partition T_p is computed by

$$f_p(\boldsymbol{x}) = \sum_{i \in T_p} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{2}$$

D. Validation Stage

Let the set of 12 preprocessed feature vectors of length D, belong to the j-th iteration, denoted by

$$X^{(j)} = \{\boldsymbol{x}_{12(j-1)+1}, \boldsymbol{x}_{12(j-1)+2}, \ldots, \boldsymbol{x}_{12j}\}. \tag{3}$$

For each set $X^{(j)}$ we have the associated set of flash codes

$$L^{(j)} = \{\ell_{12(j-1)+1}, \ell_{12(j-1)+2}, \ldots, \ell_{12j}\}, \tag{4}$$

where $\ell_i \in \{1, 2, \ldots, 12\}$ are the row/column labels as depicted in Fig. 1.

We define the sum (on all p partitions) and accumulated (after iteration j) row/column scores associated with each flash code as

$$s_r^{\mathrm{row}(j)} = \sum_{p=1}^{P} \sum_{\substack{i'=1..12j \\ \ell_{i'}=r}} f_p(\boldsymbol{x}_{i'}), \tag{5}$$

$$s_c^{\mathrm{col}(j)} = \sum_{p=1}^{P} \sum_{\substack{i'=1..12j \\ \ell_{i'}=c+6}} f_p(\boldsymbol{x}_{i'}). \tag{6}$$

where $r, c \in \{1, 2, \ldots, 6\}$.

After each iteration, a candidate pair of row/column which yields the highest scores is derived to determine the letter output:

$$\hat{r}, \hat{c} = \underset{r,c}{\mathrm{argmax}}\, \boldsymbol{M}_{rc}. \tag{7}$$

where $\boldsymbol{M}_{rc} = \boldsymbol{M}_{rc}^{(j)} = s_r^{\mathrm{row}(j)} + s_c^{\mathrm{col}(j)}$ $(r, c \in \{1, 2, \ldots, 6\})$ is the score matrix of all P3S letters. Unless stated, hereinafter we refer s_r^{row} as $s_r^{\mathrm{row}(j)}$, and s_c^{col} as $s_c^{\mathrm{col}(j)}$ for notational convenience.

III. DYNAMIC STOPPING

Our framework has the aspect that the final letter output is decided based on every individual post-stimulus trial's score $f_p(\boldsymbol{x})$ of a specific preprocessed trial vector $\boldsymbol{x}$. We exploit this aspect, as well as the computational strength, to develop the real-time stopping criteria. Instead of estimating the output only after a fixed iteration, we propose a criteria checking method to dynamically stop the flashing at any iteration, once the classifier is sufficiently confident about its decision. We propose two criteria, one based on the classifiers' scores and one based on the classifiers' votes. Our DS method thereby can be presented as follows: The flashing of a single letter spelling session is terminated after a specific iteration once both of the following criteria are met

$$\begin{cases} \theta_1 \geq \tilde{\theta}_1 \\ \theta_2 \geq \tilde{\theta}_2. \end{cases} \tag{8}$$

A. Criterion 1

Let $\Delta s^{\text{row}(j)}$ and $\Delta s^{\text{col}(j)}$ denote the margin between the highest and the second-highest scores of all rows/columns after iteration j as

$$\Delta s^{\text{row}(j)} = s_{\hat{r}}^{\text{row}} - \max_{r \neq \hat{r}}(s_r^{\text{row}}), \tag{9}$$

$$\Delta s^{\text{col}(j)} = s_{\hat{c}}^{\text{col}} - \max_{c \neq \hat{c}}(s_c^{\text{col}}). \tag{10}$$

We define θ_1 as the parameter for qualifying criterion 1 as

$$\theta_1 = \frac{\Delta s^{\text{row}} + \Delta s^{\text{col}}}{1/6 \times (\sum_r \hat{s}_r^{\text{row}} + \sum_c \hat{s}_c^{\text{col}})}, \tag{11}$$

where

$$\hat{s}_r^{\text{row}} = s_r^{\text{row}} - \min_{r'}(s_{r'}^{\text{row}}), \tag{12}$$

$$\hat{s}_c^{\text{col}} = s_c^{\text{col}} - \min_{c'}(s_{c'}^{\text{col}}). \tag{13}$$

The intuitive idea behind criterion 1 is that the ensemble accumulated margin between two highest-scored row/column candidates represents the confidence level of the classifier about its decision.

Lemma 1: The expected value of θ_1 increases with respect to j.

Proof: Lemma 1 demonstrates the validity of criterion 1, as the confidence of the classifier will increase with respect to the number of iteration j. We can easily observe that the denominator in Eq.(11) is the mean of scaled row/column scores and stays as a constant with respect to j. With regards to the nominator, we denote $\Delta s^{(j)}$ (where $\Delta s^{(j)}$ can stand for either $\Delta s^{\text{row}(j)}$ or $\Delta s^{\text{col}(j)}$) as the subtraction between the avaraged accumulated score of the target row (or column) response, $S^{+(j)}$, and the maximum score among 5 averaged accumulated scores of 5 non-target responses, $S_k^{-(j)}$, ($k = 1 \cdots 5$), after j iterations. Here, the index k of 5 non-target responses is not necessarily relevant to the non-target row/column numbering on the P3S, since we only concern about the probabilistic properties of the general target and non-target scores. $S^{+(j)}$ and $S_k^{-(j)}$ can be decomposed into random samples s_i^+ and s_i^- of size j ($i = 1 \ldots j$), respectively, drawn from some unknown distribution with the expected values given by μ^+, μ^- and variances given by σ^+, σ^-, respectively. According to the central limit theorem, regardless of the distribution of s_i^+ and s_i^-, their sum, $S^{+(j)}$ and $S_k^{-(j)}$, tend towards a Gaussian distribution

$$S^{+(j)} \sim \mathcal{N}\big(\mu^{+(j)}, \sigma^{+(j)}\big); \;\; S_k^{-(j)} \sim \mathcal{N}\big(\mu^{-(j)}, \sigma^{-(j)}\big). \tag{14}$$

As a result we can write the expected value of Δs as

$$\mathbf{E}[\Delta s^{(j)}] = \mathbf{E}[S^{+(j)} - \max_k S_k^{-(j)}] = \mu^+ - \mathbf{E}[\max_k S_k^{-(j)}]. \tag{15}$$

The variances of $S^{+(j)}$ and $S_k^{-(j)}$ decreases when their sample size of them (j) increase as

$$(\sigma^{+(j)})^2 = \text{Var}\Big[\frac{1}{j}\sum_{i=1}^{j} s_i^+\Big] = \frac{1}{j^2}\sum_{i=1}^{j}\text{Var}[s_i^+] = \frac{(\sigma^+)^2}{j},$$

$$(\sigma^{-(j)})^2 = \text{Var}\Big[\frac{1}{j}\sum_{i=1}^{j} s_i^-\Big] = \frac{1}{j^2}\sum_{i=1}^{j}\text{Var}[s_i^-] = \frac{(\sigma^-)^2}{j}.$$

The effect of the reduction in the variance of $S^{+(j)}$ does not affect $\Delta s^{(j)}$ as its mean is independent from its variance ($\mu^{+(j)} = \mu^+, \mu^{-(j)} = \mu^-$). However $S_k^{-(j)}$ does affect the mean of $\max S_k^{-(j)}$ by its variance reduction. As in [29], $\max S_k^{-(j)}$ is Gumbel-distributed with its probability density function given by

$$f_{\max S_k^{-(j)}}(x) = \frac{1}{\sigma^{-(j)}}\exp\Big(\frac{x-\mu^{-(j)}}{\sigma^{-(j)}}\Big)\exp\Big(-\exp\Big(\frac{x-\mu^{-(j)}}{\sigma^{-(j)}}\Big)\Big).$$

The corresponding expected value of $\max S_k^{-(j)}$ is

$$\mathbf{E}[\max_k S_k^{-(j)}] = \mu^{-(j)} + \gamma\,\sigma^{-(j)} = \mu^- + \gamma\,\frac{(\sigma^-)}{\sqrt{j}}, \tag{16}$$

where $\gamma \simeq 0.5772$ is the Euler-Mascheroni constant. With regards to Eq.(16), since μ^-, σ^-, and γ are all constants, then $\mathbf{E}[\max_k S_k^{-(j)}]$ decreases with respect to j. As a result, $\mathbf{E}[\Delta s^{(j)}]$ increases, hence $\mathbf{E}[\theta_1]$ also increases with respect to j.

B. Criterion 2

We define the accumulated score of row r or column c, which is validated on partition p after j iterations as

$$s_{r,p}^{\text{row}(j)} = \sum_{\substack{i'=1..12j \\ \ell_{i'}=r}} f_p(\boldsymbol{x}_{i'}), \tag{17}$$

$$s_{c,p}^{\text{col}(j)} = \sum_{\substack{i'=1..12j \\ \ell_{i'}=c+6}} f_p(\boldsymbol{x}_{i'}). \tag{18}$$

where $r, c \in \{1, 2, \ldots, 6\}$. Let $V_{rc}^{(j)}$ denote the voting count for the letter on row r and column c as

$$V_{rc}^{(j)} = \sum_p \upsilon_p^{(j)}(r, c) \tag{19}$$

where $\upsilon_p^{(j)}$ is the voting function of classifier p

$$\upsilon_p^{(j)}(r, c) = \begin{cases} 1, & \text{if } r, c = \text{argmax}_{r',c'}(s_{r',p}^{\text{row}(j)} + s_{c',p}^{\text{col}(j)}) \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Criterion 2 is characterized by the parameter θ_2 which is defined as

$$\theta_2 = \frac{\max_{r,c} V_{rc}^{(j)}}{P}. \tag{21}$$

Intuitively, criterion parameter θ_2 represents the number of constituent classifiers voting for the same output letter, which is also increasing with respect to j as similar to θ_1.

Lemma 2: For any $n \in \{1, \dots, P-1\}$, the probability $Pr(\theta_2 \geq n)$ increases with respect to j, and $\lim_{j\to\infty} \theta_2 = 1$.

Proof: Let r^* and c^* denote the true target row and column, then it can be observed that $\upsilon_p^{(j)}(r^*, c^*)$ is Bernoulli-distributed with probability ρ_p for value 1 and $1-\rho_p$ for value 0, where ρ_p is the probability of correct output letter, produced by classifier f_p, validated on one iteration. As a result, the sum of $\upsilon_p^{(j)}(r^*, c^*)$ over P classifiers, $V_{r^*c^*}^{(j)}$, has the Poisson binomial distribution with the expectation given by

$$\mathbf{E}[V_{r^*c^*}^{(j)}] = \sum_{p=1}^{P} \rho_p^{(j)}, \tag{22}$$

where $\rho_p^{(j)}$ is the accumulated probability of the correct output letter after j iterations.

As $\lim_{j\to\infty} \rho_p^{(j)} = 1 \; \forall p$ when $r = r^*, c = c^*$. Then $V_{r^*c^*}^{(j)}$ converges to P over j as

$$\lim_{j\to\infty} \mathbf{E}[V_{r^*c^*}^{(j)}] = \lim_{\rho_p^{(j)}\to 1} \sum_{p=1}^{P} \rho_p^{(j)} = P. \tag{23}$$

C. Estimation $\tilde{\theta}_1$ and $\tilde{\theta}_2$ Through Dynamic Training

We also perform an estimation for $\tilde{\theta}_1$ and $\tilde{\theta}_2$ based on the learning sets T_p. Our aim is to provide a reliable method to obtain the criteria parameters which satisfy our specific desirable accuracy. The parameters are obtained through an extensive grid search analysis with respect to the changes in $\tilde{\theta}_1$ and $\tilde{\theta}_2$.

Let $A_p(\hat{\theta}_1, \hat{\theta}_2)$ denote the accuracy when performing the validation stage on set T_p with a pair of parameters $\tilde{\theta}_1 = \hat{\theta}_1$ and $\tilde{\theta}_2 = \hat{\theta}_2$. Let $S(a)$ denote the set of parameter pairs which can attain the optimal accuracy decreased by a margin of $a\%$. The set $S(a)$ can be presented as

$$S(a) = \{(\hat{\theta}_1, \hat{\theta}_2) | A(\hat{\theta}_1, \hat{\theta}_2) \geq \max_{\substack{\theta_1' \in R_1 \\ \theta_2' \in R_2}} A(\theta_1', \theta_2') - a\}, \tag{24}$$

where

$$A(\hat{\theta}_1, \hat{\theta}_2) = \frac{1}{P} \sum_{p}^{P} A_p(\hat{\theta}_1, \hat{\theta}_2). \tag{25}$$

The tuning range of $\hat{\theta}_2$, denoted by R_2, drops within the interval $[0, \dots, 1]$, as

$$\max_{i,j} \theta_2 = \sum_{p=1}^{P} \max_{r,c} \upsilon_p^{(j)}(r, c)/P = (\sum_{p=1}^{P} 1)/P = 1. \tag{26}$$

We can also find the tuning range for $\hat{\theta}_1$, denoted by R_1, by computing $\lim_{j\to\infty} \theta_1$, but this is unnecessary. Since R_2 is found, for each specific value of $\hat{\theta}_2$ we can perform the search by iteratively increasing $\hat{\theta}_1$ until the maximum accuracy is achieved. The step for iteratively increasing $\hat{\theta}_1$ and $\hat{\theta}_2$, denoted by η_1 and η_2 respectively, is freely chosen (i.e., $\eta_2 = 0.25$, and $\eta_1 = 0.2$).

To obtain the least possible iterations (for fastest performance), a unique corresponding pair of $\tilde{\theta}_1, \tilde{\theta}_2$ can be achieved by

$$\tilde{\theta}_1, \tilde{\theta}_2 = \operatorname*{argmin}_{(\hat{\theta}_1, \hat{\theta}_2)\in S} \{A(\hat{\theta}_1, \hat{\theta}_2)\}. \tag{27}$$

The algorithm of the iterative grid search method is presented in Algorithm 1.

Algorithm 1 Iterative Grid Search for $\tilde{\theta}_1, \tilde{\theta}_2$

Input: a
Output: $\tilde{\theta}_1, \tilde{\theta}_2$
Initialization: η_1, η_2
$\hat{\theta}_1 \leftarrow 0, \hat{\theta}_2 \leftarrow 0, A_{max} \leftarrow 0$, STOP_FLAG $\leftarrow 0$, $S \leftarrow \emptyset$
while STOP_FLAG $\neq 1$ **do**
  for $\hat{\theta}_2 = 0:1$ (step η_2) **do**
    Compute and save $A(\hat{\theta}_1, \hat{\theta}_2)$
    if $A(\hat{\theta}_1, \hat{\theta}_2) > A_{max}$ **then** $A_{max} \leftarrow A(\hat{\theta}_1, \hat{\theta}_2)$
    end if
  end for
  if $A(\hat{\theta}_1, 0) = A_{max}$ and $A(\hat{\theta}_1, 0) = A(\hat{\theta}_1 - \eta_1, 0)$ **then**
    STOP_FLAG $\leftarrow 1$;
  end if
  $\hat{\theta}_1 \leftarrow \hat{\theta}_1 + \eta_1$
end while
for each $\hat{\theta}_1, \hat{\theta}_2$ **do**
  if $A(\hat{\theta}_1, \hat{\theta}_2) \geq A_{max} - a$ **then** add $(\hat{\theta}_1, \hat{\theta}_2)$ to S
  end if
end for
Search for $\hat{\theta}_1, \hat{\theta}_2$ in S
return $\hat{\theta}_1, \hat{\theta}_2$

IV. Adaptive Learning

Adaptive learning (AL) is a method of continually incorporating new test samples that have been classified during the on-line experiments into the existing classifier. This method can make the classifier that was trained on different subjects to get adapted with the ongoing subject's patterns, thereby increase the classification accuracy. The arising problem of AL is that we do not know if the new samples' label are correctly classified, as if we wrongly label them they can negatively affect the existing classifier. As a result, the effects of AL are heavily governed by the performance of the DS method. The influence and relationship of DS and AL are further discussed in Section V-F and Section V-G.

There have been various approaches on adaptive SVM [25], [26], [30], [31]. We employ the method, called incremental SVM (ISVM) that was inspired by [25] because it was well-developed to integrate new training samples without re-solving the quadratic program over the whole training set. ISVM is an incremental learning method with *discriminative* manner, i.e., analytically tuning the existing classifier using new data sample-by-sample. As compared to the *population-based* methods of the papers in Table IV (mostly used Bayesian approach), ISVM has a significantly better generalization ability and accuracy [32]. especially

Algorithm 2 On-Line Incremental Learning Update for Continual P300 Spelling Sessions

Initialization: $k \leftarrow 0, \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_{\mathcal{L}}, b \leftarrow b_{\mathcal{L}}$
while USER_STOP $\neq 1$ **do**
    Implement flashing paradigm of the k-th session
    Implement Dynamic Stopping
    Obtain $s, \tilde{i}_s, \tilde{j}_s$ using solution $(\boldsymbol{\alpha}, b)$
    Input: USER_STOP (1 for stop)
    Assign new labels for $y_{\mathcal{T},t}$ according to $\tilde{i}_s, \tilde{j}_s$
    Add new $N_2 = s \times 12$ samples to $\mathcal{T} = \{(\boldsymbol{x}_{\mathcal{T},t}, y_{\mathcal{T},t})\}_{t=1}^{N_2}$
    Increment solution $(\boldsymbol{\alpha}, b) \leftarrow \text{ISVM}(\mathcal{L}, \boldsymbol{\alpha}, b, \mathcal{T})$
end while

when adjusting the existing classifier with some few newly-classified samples. However, one drawback of ISVM is the complicated algorithm and time-consuming learning time.

Suppose we need to integrate the set of N_1 newly-classified test trials, denoted by $\mathcal{T}$, into the existing SVM solutions that were learned from p sets (T_p) of N training samples. Cauwenberghs and Poggio [25] proposed an incremental method by adding new samples from $\mathcal{T}$, one at a time, into the existing solution $\{\boldsymbol{\alpha}_p, b_p\}$, by analytically tuning the solution to retain the Kuhn-Tucker conditions on all samples. In short, the method attempts to find the final solution $\{\boldsymbol{\alpha}_{p,\text{new}}, b_{p,\text{new}}\}$ of the set $T_p \cup \mathcal{T}$ that can be expressed in terms of $\{\boldsymbol{\alpha}_p, b_p\}$ and the samples of $\mathcal{T}$ as

$$(\boldsymbol{\alpha}_{p,\text{new}}, b_{p,\text{new}}) = \text{ISVM}(T_p, \boldsymbol{\alpha}_p, b_p, \mathcal{T}). \tag{28}$$

In this paper, we assign new labels for new validation samples of $\mathcal{T}$ based on the letter decision of the whole previous letter session, rather than the individual output of each vector. The benefit of this approach is that once the letter of the previous session is correctly classified, it will increase the next session's performance. However, a misclassified letter may negatively affect the accuracy of later sessions. This effect will be discussed in Section V-F.

The number of samples in $\mathcal{T}$ may vary from 1 to 15 iterations (corresponding to 12 samples to 180 samples), depending on the DS performance of the previous session. Algorithm 2 presents the detailed updating process after each spelling session. The computational time required for this process will be further discussed in Section V-B.

V. Experiments and Discussion

A. Signal Preprocessing

A single post-stimulus signal, associated with a flash, is 666.67 millisecond long, and consists of 160 samples per channel. We employ a 4-th order Chebyshev band-pass filter at [0.1 20] Hz for each individual trial. After decimation, each post-stimulus preprocessed vector $\boldsymbol{x}_i \forall i$ (as mentioned in Section II) has the length of 140 samples (14 samples $\times$ 10 channels).

During the learning stage, each decimated vector $\boldsymbol{x}_i$ is normalised based on all N vectors in the learning set T_p. During the validation stage, as we do not have any knowledge on the normalisation scale of a new input vector $\boldsymbol{x}$, it is important for the classifier to memorize the normalisation parameters $\bar{\boldsymbol{x}}$ and $\boldsymbol{\sigma}_{\boldsymbol{x}}$ obtained from the learning stage.

B. Influence of Partitioning on Adaptive Learning

Fig. 4a and Fig. 4b illustrate the evolution of the time needed for updating the existing classifiers with respect to the number of new learning samples (from 12 samples to 180 samples). Each plot consists of 4 different line graphs associated to different numbers of existing samples in the classifier pool (i.e., 540, 1080, 1620, and 2160). Without partitioning, the AL updating time drastically increases with respect to the number of new learning samples, as shown in Fig. 4a. For instance, to integrate 180 new samples into 1080 existing samples, we need roughly 5 seconds. But with the same number of new samples it takes over 17 seconds to integrate them into 2160 existing samples. However, just by dividing the 2160 existing samples into 3 partitions (720 each) as shown in Fig. 4b, we can drastically reduce the AL time from 17 seconds to 8 seconds. This proves the huge benefit of partitioning in terms of AL time reduction.

One desired job in our proposed method is to determine the appropriate number of samples and partitions of the existing ensemble classifier to best fit the AL stage. Typically, a P3S gives the subjects a short break (from 5 to 10 seconds) between letter sessions to help them relax and find the next target letter to focus. This break time can be exploited by the AL stage to update the classifier between sessions. As a result each AL stage must not last too long (i.e., over 10 seconds). Assuming the worst performance is performed by the DS stage (i.e., we need the maximum of 15 iterations to output the decision), we plot the average time needed (in seconds) to integrate 180 new samples (15 iterations) into 1080, 2160, and 3240 existing samples of the classifier, as shown in Fig. 4c. It is observed from the figure that using 3240 existing samples is not an option, since it takes longer than 10 seconds in all partitioning schemes. On the other hand, when updating 1080 and 2160 existing samples (with partitioning), the AL time is below 5 seconds and 10 seconds, respectively, which satisfies our break time criterion. We choose 2160 to be the fixed number of learning samples, as more existing samples undoubtedly result in better performance. As a result, we make further analysis on the accuracies and iterations of different partitioning schemes using 2160 samples, as described in Table I.

C. Data Sets

In this paper, we conduct our analysis on the Akimpech dataset [33]. As compared to the other public P3S datasets, Akimpech contains a large number of subjects (26, as compared to 1 of [34], 2 of [35], or 9 of [14]). This advantage allows us to carry out our extensive numerical analysis with strong probabilistic aspect. Moreover, each subject in the Akimpech dataset also conducted a large number of experiments (27-39 letters). Each letter session is 15-iteration long.

We use the data of 12 *learning subjects* (namely, ACS, APM, ASG, ASR, CLL, DCM, DLP, LGP, ELC, JCR, FSZ,

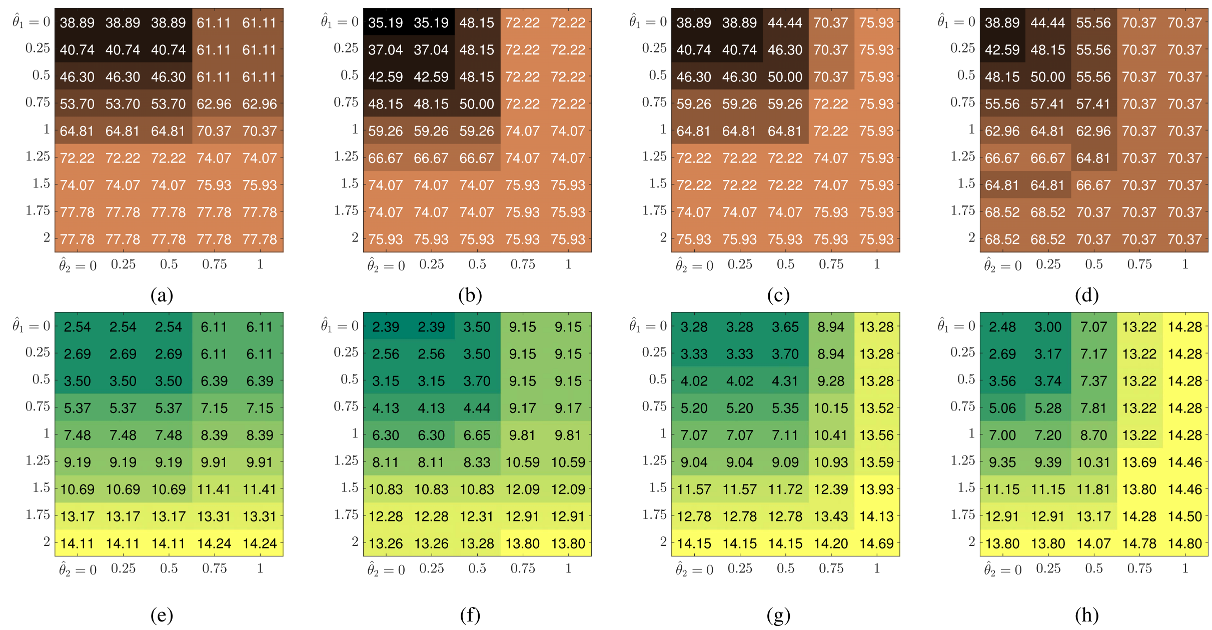

Fig. 3. The accuracies and corresponding number of iterations of 4 schemes conducted on the learning set with different values of dynamic stopping parameters $\hat{\theta}_1$ and $\hat{\theta}_2$. (a) ENS-2 accuracy. (b) ENS-3 accuracy. (c) ENS-4 accuracy. (d) ENS-6 accuracy. (e) ENS-2 iterations. (f) ENS-3 iterations. (g) ENS-4 iterations. (h) ENS-6 iterations.

GCE) of the Akimpech dataset to implement the learning process. The first phase is to construct four classifiers (ENS-2, ENS-3, ENS-4, ENS-6) from the first six subjects (ACS, APM, ASG, ASR, CLL, and DCM), as presented in Table I. The data of six other subjects (with 9 letters each) are used to evaluate the DS parameters $\tilde{\theta}_1$ and $\tilde{\theta}_2$.

The resulting parameters are then used to validate the large test set of the other 14 *validation subjects* (FSZ, GCE, ICE, IZH, JLD, JLP, JMR, JSC, JST, LAC, LAG, PGA, WFG, XCL), with 12 to 24 letters per subject.

D. Grid Search Result

Fig. 3 illustrates the expected accuracy and number of iterations validated from the learning stage using different values of DS parameters $\tilde{\theta}_1$ and $\tilde{\theta}_2$. Generally, the numbers of iterations vary very slightly given a specific pair of DS parameters over the 4 schemes. Similarly, the expected accuracies heat maps of the 4 schemes are also similar as they never reach above 80%. From Fig. 3, we extract the searching results for $\tilde{\theta}_1$ and $\tilde{\theta}_2$ given the expected accuracy reduction a (as in Algorithm 1), as shown in Table II.

TABLE II
THE GRID SEARCH RESULTS OF 4 CLASSIFICATION SCHEMES USING 3 INPUT VALUES OF $a = 0$, $a = 5$, AND $a = 10$ AS IN ALGORITHM 1

Scheme	a	$\tilde{\theta}_1$	$\tilde{\theta}_2$	Expected Accuracy	Expected Iteration
ENS-2	0	1.75	0.50	77.78	13.17
	5	1.25	0.75	74.07	9.91
	10	1.00	0.75	70.37	8.39
ENS-3	0	1.50	0.75	75.93	11.70
	5	0.50	0.75	72.22	9.15
	10	1.25	0.25	66.67	8.11
ENS-4	0	0.50	1.00	75.93	13.28
	5	1.25	0.25	72.22	9.04
	10	0.25	0.75	70.37	8.94
ENS-6	0	0.50	0.75	70.37	12.32
	5	1.25	0.00	66.67	9.35
	10	1.00	0.00	62.96	7.00

E. Validation Stage Performance

In our paper, the term *Accuracy* for each subject is simply computed as

$$Accuracy = \frac{\text{correct classified letters}}{\text{total letters spelled}},$$

and the term *Iteration* of a subject is the number of iterations taken averaged over all his/her spelling letters. The term *ITR*, which stands for information transfer rate, is the harmonized metric between *Accuracy* and *Time*, and is calculated by

$$ITR = \frac{\log_2 36 + A \log_2 \ A + (1 - A) \log_2 (1 - A)}{Iteration \times ISI \times 12/60},$$

where A is the accuracy and ISI is the inter-stimulus interval (seconds). Finally, the term *MaxTime* is defined as the maximum AL time (in seconds) for a subject taken over all the spelling sessions. More specifically, for each subject we take the maximum AL time among multiple letters spelled by that subject, then we take the average of maximum AL time of all subjects. Using those 4 evaluation metrics, we present the average *Accuracy*, average *Iteration*, average *ITR*, and average *MaxTime* taken on all validation subjects (as described in Section V-C) in Table III.

It should be noticed that given the existing 2160 samples of the classifier at the beginning, we should not employ AL for all approaching testing sessions, as the classifier size will be excessively accumulated which leads to increasing AL time. Since each subject has 12 to 24 letters to validate, we only use their first 0, 2, 4, 6, or 8 new letter sessions for the AL stage, then analyze the changes in accuracies and iterations

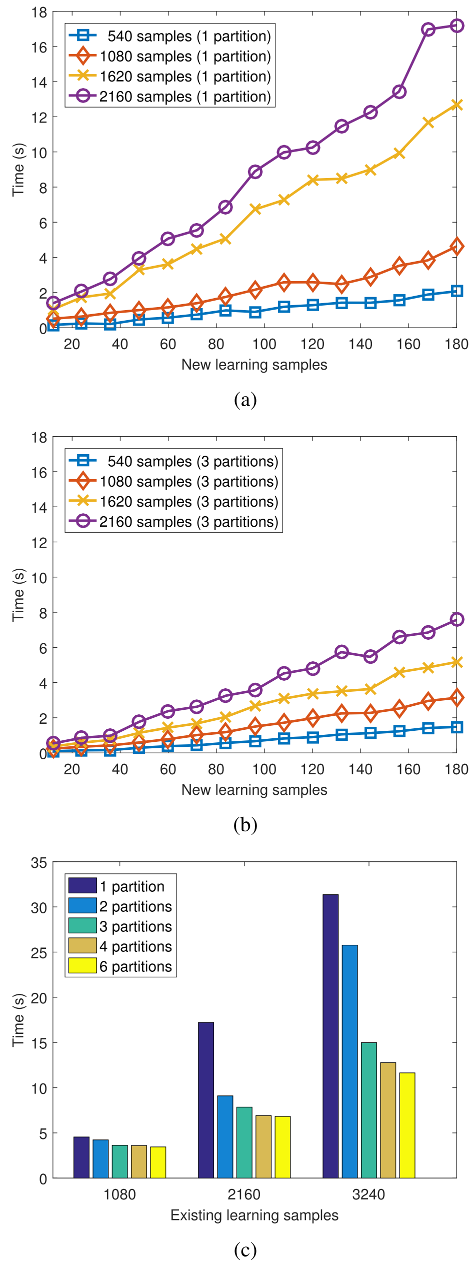

Fig. 4. Time needed to perform adaptive learning stage with various settings. (a) Time vs. new learning samples, with no partitioning. 4 lines indicate 4 different numbers of existing samples in the available classifier solution. (b) Time vs. new learning samples, with partitioning. 4 lines indicate 4 different numbers of existing samples in the available classifier solution. (c) Time vs. existing samples in the classifier solution and classifier partitions.

TABLE III
CLASSIFICATION RESULT TAKEN AVERAGE ON ALL VALIDATION SUBJECTS

$a = 0$

Letters for AL		0	2	4	6	8
ENS-2	*Accuracy*	79.20	90.98	92.74	94.89	95.96
	Iteration	12.71	11.31	10.47	9.87	9.25
	ITR	7.57	10.78	12.06	13.36	14.57
	MaxTime	0.00	6.52	9.82	11.27	13.18
ENS-3	*Accuracy*	83.11	93.99	96.27	96.18	96.58
	Iteration	11.87	9.23	8.22	7.84	7.61
	ITR	8.78	14.03	16.50	17.27	17.94
	MaxTime	0.00	3.48	6.01	6.92	8.52
ENS-4	*Accuracy*	81.89	89.90	90.87	90.04	89.46
	Iteration	12.16	6.63	5.50	4.90	4.60
	ITR	8.36	18.01	22.13	24.43	25.73
	MaxTime	0.00	3.45	4.79	4.92	5.97
ENS-6	*Accuracy*	82.75	88.30	86.10	83.93	81.83
	Iteration	12.73	6.14	5.14	4.50	4.29
	ITR	8.12	18.84	21.54	23.55	23.67
	MaxTime	0.00	3.50	4.95	5.42	6.24

$a = 5$

Letters for AL		0	2	4	6	8
ENS-2	*Accuracy*	75.72	86.04	90.58	91.21	92.32
	Iteration	9.84	8.18	7.81	6.80	6.47
	ITR	9.07	13.51	15.50	18.02	19.36
	MaxTime	0.00	4.24	7.28	9.85	10.23
ENS-3	*Accuracy*	80.76	84.98	85.09	85.43	83.21
	Iteration	8.78	5.81	4.70	4.19	3.92
	ITR	11.31	18.63	23.08	26.07	26.64
	MaxTime	0.00	2.12	3.27	4.08	4.15
ENS-4	*Accuracy*	76.30	84.70	87.83	89.25	90.12
	Iteration	8.95	6.81	6.47	6.09	6.02
	ITR	10.10	15.80	17.71	19.35	19.92
	MaxTime	0.00	2.81	4.65	5.95	7.15
ENS-6	*Accuracy*	75.91	86.00	88.63	87.15	87.24
	Iteration	8.75	6.75	6.59	6.20	6.01
	ITR	10.25	16.37	17.67	18.24	18.85
	MaxTime	0.00	3.12	4.60	6.31	7.26

$a = 10$

Letters for AL		0	2	4	6	8
ENS-2	*Accuracy*	71.48	79.86	84.53	86.46	87.41
	Iteration	8.26	6.66	6.12	5.26	5.28
	ITR	9.83	14.63	17.53	21.20	21.52
	MaxTime	0.00	3.25	5.70	8.10	8.94
ENS-3	*Accuracy*	77.94	88.82	88.96	90.09	90.94
	Iteration	8.56	7.17	6.53	6.09	6.04
	ITR	10.93	16.30	17.95	19.68	20.18
	MaxTime	0.00	2.86	4.70	5.61	6.62
ENS-4	*Accuracy*	76.18	77.67	70.89	68.93	69.51
	Iteration	8.31	3.83	3.29	2.98	2.90
	ITR	10.85	24.30	24.37	25.71	26.78
	MaxTime	0.00	1.94	2.33	2.48	2.62
ENS-6	*Accuracy*	68.19	78.87	73.26	76.25	76.38
	Iteration	6.63	5.61	4.94	4.72	4.56
	ITR	11.35	17.02	17.12	19.13	18.63
	MaxTime	0.00	2.17	3.34	5.51	5.86

with respect to the numbers of new letters learned. As shown in Table III, for all input values of a, there are huge leaps in accuracies when employing the AL process. For instance, when *Letters for AL* equals 2 ($a = 0$) we can increase the accuracy from around 80% to around 90% in all 4 schemes ENS-2, ENS-3, ENS-4 and ENS-6. However when increasing the letters for AL, the accuracies increase more slowly and the iterations also drop more slowly in all schemes (for $a = 0$ and $a = 5$), but the AL time increases much faster. This effect is not desirable in our real-time system. For that reason we only need to use the AL stage for a very first few letters (optimally 2 to 4 letters) of a new subject, which can balance between the accuracy and AL time.

F. Influence of Expected Accuracy Parameter a of the Grid Search Algorithm

When using a low expected accuracy parameter ($a = 10$), it is observed from Table III that the accuracies do not increase but fluctuate (in ENS-4 and ENS-6) with respect to the number of AL letters. The explanation for this effect is that the performance is hugely influenced by the correctness

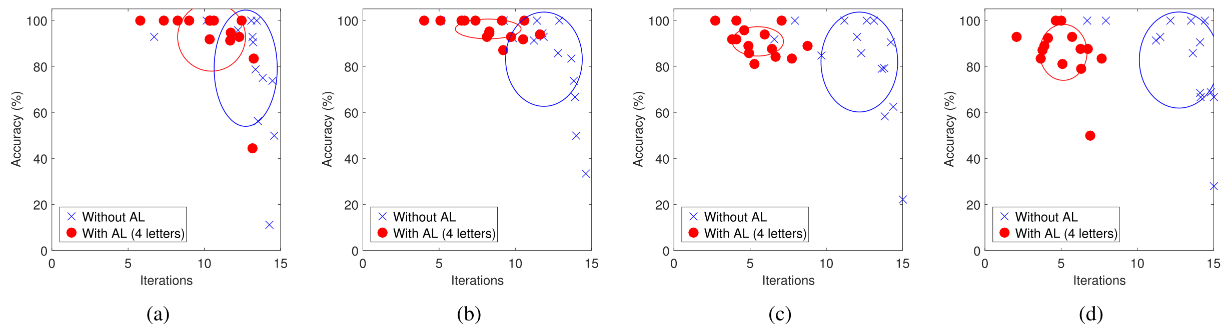

Fig. 5. The subject-wise results for all test subjects using different input values of dynamic stopping parameters $\hat{\theta}_1$ and $\hat{\theta}_2$. Each individual point represents the performance of one subject. The ellipses represents the average area of each scheme, with the centers of the ellipses are the mean and the radii are the standard deviations of the accuracy and iterations on the whole scheme. (a) ENS-2. (b) ENS-3. (c) ENS-4. (d) ENS-6.

of the previous classified letters. A wrongly-classified letter will negatively affect the existing classifier and cause the deterioration in accuracies of the following sessions. The safe tuning range for a is therefore quite narrow (from 0 to 5). If the highest accuracy is desired, $a = 0$ is the optimal choice, whereas $a = 5$ will slightly decrease the accuracies but also reduce the number of iterations significantly. This choice depends on user-purposed system specifications.

G. Influence of DS and AL on the Performance of Each Subject

Fig. 5 plots the subject-wise performance of all schemes using 4 AL letters with $a = 0$. Each individual point on the plots represents the classification performance (accuracy and iteration) of one subject. The ellipses are drawn to represent the average region (of all points) of a specific setting on the plot. As shown, we achieve a significant reduction in iterations in all schemes. The regions of AL points are narrower, implying that the performances of all subjects will converge to a specific accuracy and iteration given a fixed setting. There are points (without AL) over the subplots which have extremely poor accuracies (below 60%). This problem is caused by the mismatch between the brain patterns of the validation subject and the learning subjects. However with adaptive learning, this problem is solved.

H. Comparisons with Related Studies

We have re-implemented the method used in other related studies [11], [16], [17], [23], [36]–[38], using the equivalent classifier and dataset as specified in our work. Our target is that we want to provide a most reliable comparison between our work with the others. The comparison is shown in Table IV.

Overall we outperformed the others in both accuracy and iterations. Tao *et al.* [37] took the sum over the last 3 iterations of row/column candidates and compared them with the threshold $N \times d$, where N is the free parameter and d is the average distance of target responses to the classifier solution, derived from the training set. We employed their method using $N = 1$. Jin *et al.* [38] used 2 dynamic stopping parameters. N_1 is the number of consecutive iterations which output the same letter, and N_2 is the beginning iteration to start checking N_1. As suggested in [38] we conducted their method using $N_1 = 2$ and $N_2 = 3$. More recently, a probabilistic approach for DS stage was used in [11], [16], [17], [23], and [36]. Although those mentioned studies have different research problems, they however were employed using the similar DS method first proposed by [23]. This method calculates the probability of each letter candidate after each iteration via a Bayesian updating basis and compares them to the threshold t. To make a comparison with our method we also re-conducted their approach in our schemes, using $t = 0.95$.

TABLE IV
COMPARISON OF OUR STUDY AND RELATED STUDIES, CONDUCTED ON THE SUBJECT-INDEPENDENT BASIS

Method	Setting	Accuracy (%)	Iterations
[11], [16], [17], [23], [36]	ENS-2	60.85	6.56
	ENS-3	64.07	4.84
	ENS-4	65.24	4.90
	ENS-6	65.65	5.06
[37]	ENS-2	52.63	3.00
	ENS-3	57.90	3.00
	ENS-4	52.55	3.05
	ENS-6	61.23	5.32
[38]	ENS-2	48.95	9.36
	ENS-3	56.41	8.82
	ENS-4	47.56	9.45
	ENS-6	51.45	8.64
Our method	ENS-2	95.96	9.25
	ENS-3	96.58	8.52
	ENS-4	90.87	5.50
	ENS-6	88.30	3.50

VI. CONCLUSION

In this paper we have provided a complete solution to integrate ensemble classifier, dynamic stopping, and adaptive learning based on the SVM scheme to boost the performance of event-related potential BCI, especially on the subject-independent basis. Our experimental results suggest that, instead of achieving the average (taken over 4 schemes with $a = 0$) accuracy of 75.00% with 12.62 iterations for each spelling session, we can boost the accuracy up to 91.26% with just 6.78 iterations. The trade-off for this improvement is the huge computational resources for updating the existing classifier. However, with thorough benchmarking and analysis, we also propose a method to achieve the most appropriate dynamic stopping parameters and adaptive learning settings. As a consequence, under our control the additional computation does not affect the system since the free time between letter sessions is efficiently exploited. To emphasize the merit of our proposal, we also re-implement the methods from related works using the same subject-independent setting as in ours. Our method outperformed most of the related studies in terms of accuracy. With an appropriate selection on the ensemble scheme, the experimental speed is also significantly boosted. For future work, a more efficient performance can be achieved by employing a language model into our framework, or by conducting a selection process of EEG channels. Our source code is provided online at *https://github.com/voanhkha/Subject-Independent-ERP-based-BCI*.

References

[1] L. A. Farwell and E. Donchin, "Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalogr. Clin. Neurophysiol.*, vol. 70, no. 6, pp. 510–523, Dec. 1988.

[2] S. Sutton, M. Braren, J. Zubin, and E. R. John, "Evoked-potential correlates of stimulus uncertainty," *Science*, vol. 150, no. 3700, pp. 1187–1188, 1965.

[3] T. W. Picton, "The P300 wave of the human event-related potential," *J. Clin. Neurophysiol.*, vol. 9, no. 4, pp. 456–479, 1992.

[4] H. Cecotti, M. P. Eckstein, and B. Giesbrecht, "Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 11, pp. 2030–2042, Nov. 2014.

[5] G. Dornhege, B. Blankertz, G. Curio, and K. R. Müller, "Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 993–1002, Jun. 2004.

[6] H.-I. Suk and S.-W. Lee, "A novel Bayesian framework for discriminative feature extraction in brain-computer interfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 286–299, Feb. 2013.

[7] T. Yu, Z. Yu, Z. Gu, and Y. Li, "Grouped automatic relevance determination and its application in channel selection for P300 BCIs," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 6, pp. 1068–1077, Nov. 2015.

[8] E. Yin, T. Zeyl, R. Saab, T. Chau, D. Hu, and Z. Zhou, "A hybrid brain–computer interface based on the fusion of P300 and SSVEP scores," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 4, pp. 693–701, Jul. 2015.

[9] G. Townsend, J. Shanahan, D. B. Ryan, and E. W. Sellers, "A general P300 brain–computer interface presentation paradigm based on performance guided constraints," *Neurosci. Lett.*, vol. 531, no. 2, pp. 63–68, 2012.

[10] J. Jin, E. W. Sellers, Y. Zhang, I. Daly, X. Wang, and A. Cichocki, "Whether generic model works for rapid ERP-based BCI calibration," *J. Neurosci. Methods.*, vol. 212, no. 1, pp. 94–99, Jan. 2013.

[11] P. J. Kindermans, H. Verschore, and B. Schrauwen, "A unified probabilistic approach to improve spelling in an event-related potential-based brain–computer interface," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 10, pp. 2696–2705, Oct. 2013.

[12] P. J. Kindermans, M. Tangermann, K.-R. Müller, and B. Schrauwen, "Integrating dynamic stopping, transfer learning and language models in an adaptive zero-training ERP speller," *J. Neural Eng.*, vol. 11, no. 3, p. 035005, Jun. 2014.

[13] W. Speier, C. Arnold, and N. Pouratian, "Integrating language models into classifiers for BCI communication: A review," *J. Neural Eng.*, vol. 13, no. 3, p. 031002, 2016.

[14] U. Hoffmann, J.-M. Vesin, T. Ebrahimi, and K. Diserens, "An efficient P300-based brain–computer interface for disabled subjects," *J. Neurosci. Methods*, vol. 167, no. 1, pp. 115–125, 2008.

[15] E. W. Sellers and E. Donchin, "A P300-based brain–computer interface: Initial tests by ALS patients," *Clin. Neurophysiol.*, vol. 117, no. 3, pp. 538–548, 2006.

[16] B. O. Mainsah *et al.*, "Increasing BCI communication rates with dynamic stopping towards more practical use: An ALS study," *J. Neural Eng.*, vol. 12, no. 1, p. 016013, 2015.

[17] J. M. Clements, E. W. Sellers, D. B. Ryan, K. Caves, L. M. Collins, and C. S. Throckmorton, "Applying dynamic data collection to improve dry electrode system performance for a P300-based brain–computer interface," *J. Neural Eng.*, vol. 13, no. 6, p. 066018, 2016.

[18] P.-J. Kindermans, D. Verstraeten, and B. Schrauwen, "A Bayesian model for exploiting application constraints to enable unsupervised training of a P300-based BCI," *PLoS ONE*, vol. 7, no. 4, p. e33758, 2012.

[19] N. Jrad *et al.*, "sw-SVM: Sensor weighting support vector machines for EEG-based brain–computer interfaces," *J. Neural Eng.*, vol. 8, no. 5, p. 056004, 2011.

[20] A. Rakotomamonjy and V. Guigue, "BCI competition III: Dataset II-ensemble of SVMs for BCI P300 speller," *IEEE Trans. Biomed. Eng.*, vol. 55, no. 3, pp. 1147–1154, Mar. 2008.

[21] Y. Li, C. Guan, H. Li, and Z. Chin, "A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system," *Pattern Recognit. Lett.*, vol. 29, no. 9, pp. 1285–1294, 2008.

[22] M. Spuler, W. Rosenstiel, and M. Bogdan, "Adaptive SVM-based classification increases performance of a MEG-based Brain-Computer Interface (BCI)," in *Artificial Neural Networks and Machine Learning*. New York, NY, USA: Springer-Verlag, 2012, pp. 669–676.

[23] C. S. Throckmorton, K. A. Colwell, D. B. Ryan, E. W. Sellers, and L. M. Collins, "Bayesian approach to dynamically controlling data collection in P300 spellers," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, no. 3, pp. 508–517, May 2013.

[24] B. O. Mainsah, L. M. Collins, and C. S. Throckmorton, "Using the detectability index to predict P300 speller performance," *J. Neural Eng.*, vol. 13, no. 6, p. 066007, 2016.

[25] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2000, pp. 388–394.

[26] C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 4. 2003, pp. 2685–2690.

[27] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 144–152.

[28] L. O. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 301–320.

[29] A. Fougeres, "Multivariate extremes," in *Extreme Values in Finance, Telecommunications, and the Environment, Monographs on Statistics and Applied Probability*, vol. 99. London, U.K.: Chapman & Hall, 2004.

[30] M. Palaniswami and A. Shilton, "Adaptive support vector machines for regression," in *Proc. 9th Int. Conf. Neural Inf. Process. (ICONIP)*, vol. 2. Nov. 2002, pp. 1043–1049.

[31] R. Herbrich and J. Weston, "Adaptive margin support vector machines for classification," in *Proc. 9th Int. Conf. Artif. Neural Netw. (ICANN)*, vol. 2. 1999, pp. 880–885.

[32] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, Jan. 2018.

[33] O. Yanez-Suarez, L. Bougrain, C. Saavedra, E. Bojorges, and G. Gentiletti. (2011). *P300-Speller Public-Domain Database*. [Online]. Available: http://akimpech.izt.uam.mx/p300db

[34] M. Kaper, P. Meinicke, U. Grossekathoefer, T. Lingner, and H. Ritter, "BCI competition 2003-data set IIb: Support vector machines for the P300 speller paradigm," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1073–1076, Jun. 2004.

[35] B. Blankertz *et al.*, "The BCI competition III: Validating alternative approaches to actual BCI problems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 14, no. 2, pp. 153–159, Jun. 2006.

[36] B. O. Mainsah, K. A. Colwell, L. M. Collins, and C. S. Throckmorton, "Utilizing a language model to improve online dynamic data collection in P300 spellers," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 837–846, Jul. 2014.

[37] L. Tao, G. Leslie, G. Shangkai, and H. Bo, "An online brain–computer interface using non-flashing visual evoked potentials," *J. Neural Eng.*, vol. 7, no. 3, p. 036003, 2010.

[38] J. Jin *et al.*, "An adaptive P300-based control system," *J. Neural Eng.*, vol. 8, no. 3, p. 036006, Apr. 2011.