Adaptive Compressed Sensing Architecture in Wireless Brain-Computer Interface

Aosen Wang[1], Chen Song[1], Zhanpeng Jin[2] and Wenyao Xu[1]
[1]CSE Dept., SUNY at Buffalo, NY, USA
[2] ECE Dept., SUNY at Binghamton, NY, USA
{aosenwan, csong5, wenyaoxu}@buffalo.edu, zjin@binghamton.edu

ABSTRACT

Wireless sensor nodes advance the brain-computer interface (BCI) from laboratory setup to practical applications. Compressed sensing (CS) theory provides a sub-Nyquist sampling paradigm to improve the energy efficiency of electroencephalography (EEG) signal acquisition. However, EEG is a structure-variational signal with time-varying sparsity, which decreases the efficiency of compressed sensing. In this paper, we present a new adaptive CS architecture to tackle the challenge of EEG signal acquisition. Specifically, we design a dynamic knob framework to respond to EEG signal dynamics, and then formulate its design optimization into a dynamic programming problem. We verify our proposed adaptive CS architecture on a publicly available data set. Experimental results show that our adaptive CS can improve signal reconstruction quality by more than 70% under different energy budgets while only consuming 187.88 nJ/event. This indicates that the adaptive CS architecture can effectively adapt to the EEG signal dynamics in the BCI.

1. INTRODUCTION

The brain-computer interface (BCI) is a rapidly growing technology with highly diverse applications [1]. Wireless implanted and wearable EEG sensors enable the mobility of BCI, yet are also limited by energy inefficiency because of the high-sampling rate of EEG signals and power-hungry wireless communication. The compressed sensing (CS) theory provides a promising solution to this challenge [2]. The CS samples the original signal by the sub-Nyquist rate, which is proportional to its intrinsic information. The reconstruction algorithms accurately recover the original signal with provably high probability. However, existing CS architectures pre-assume that the signal sparsity is known and consistent, which lacks adaptivity to cope with structure-variational signals, such as EEG.

Recently, there are a few related works about the adaptivity of CS frameworks. Malloy *et al.* proposed a compressive adaptive sense and search (CASS) [3] method to update the matrix on each iteration, which needs to obtain the signal sparsity ahead of time. Wang *et al.* [4] presented an algorithm to adjust the compressive measurement number dynamically. This method traversed possible measurement dimensions to compare the signal reconstruction quality and determine the optimal compression ratio, which is computationally intensive. However, these works focused on adaptive algorithms of the signal reconstruction in CS, and to date there is no sophisticated research about adaptive CS architecture.

In this paper, we present an adaptive CS architecture for the BCI applications. We introduce the concept of dynamic knob to specifically deal with the structural-variational EEG signals. Because of the design considerations of dynamic knob, i.e., adaptive controlling and ultra-low power design, we propose a template-based structure of dynamic knob in CS using a supervised learning algorithm. Specifically, we design an support vector machine (SVM) based cascaded signal analyzer to recognize the signal structure. We also model the design space in a close form and formulate it into a dynamic programming problem. We benchmark the performance of proposed adaptive CS architecture with the real EEG signals from publicly available data set [5]. The evaluation results indicate that compared with the traditional CS [2], the adaptive CS architecture can improve signal reconstruction quality by more than 70% with the energy consumption of 187.88 nJ/event under any given energy constraint.

The remainder of this paper is organized as follows: Section 2 introduces the background of Quantized CS theory. Our proposed adaptive CS architecture is discussed along with an optimization problem formulation of dynamic knob design in Section 3, and Section 4 presents the related experiments and evaluations. The conclusion and future work are described in Section 5.

2. QUANTIZED COMPRESSED SENSING

Compressed sensing theory is a new emerging analog-to-information sampling scheme for the signals that are known to be sparse or compressible under certain basis. We assume that the x is an N-dimension vector and sampled using M-measurement vector y:

$$y = \Phi x, \tag{1}$$

where $\Phi \in R^{M \times N}$ is the sensing array that models the linear encoding, and M is defined as the sampling rate in N-dimensional CS. The elements in Φ are either Gaussian random variables or Bernoulli random variables. Because of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
DAC'15, June 07 - 11, 2015, San Francisco, CA, USA
Copyright 2015 ACM 978-1-4503-3520-1/15/06$15.00
http://dx.doi.org/10.1145/2744769.2744792

$M << N$, the formulation in Eq. (1) is undetermined, and the signal x can not be uniquely retrieved from sensing array Φ and measurements y. However, under a certain sparsity-inducing basis $\Psi \in R^{N \times N}$, the signal x can be represented by a set of sparse coefficients $u \in R^N$:

$$x = \Psi u, \tag{2}$$

that is, the coefficient u, under the transformation Ψ, has few non-zero elements. Therefore, based on Eq. (1) and (2), the sparse vector, u, can be represented as follows:

$$y = \Phi \Psi u = \Theta_{M \times N} u, \tag{3}$$

where $\Theta_{M \times N} = \Phi \Psi$ is an $M \times N$ matrix, called the measuring matrix. In practical applications, original signals are analog in nature and need to be quantized before transmitting and digital processing. Therefore, the compressed signal, y, should be processed by a quantization model [6] formulated as follows:

$$\widehat{y} = Q_b(y), \tag{4}$$

where $Q_b(.)$ is the quantization function , and $\widehat{y}$ is the quantized representation of y with b bits. Due to the prior knowledge that the unknown vector u is sparse, u can be estimated based on $\widehat{y}$ using the ℓ_0 minimization formulation as follows:

$$\widehat{u} = min\ \|u\|_0 \quad s.t. \quad \|\widehat{y} - \Theta u\| < \epsilon, \tag{5}$$

where ϵ is the reconstruction error margin. The formulation in Eq. (5) is a determined system with unique solutions. However, ℓ_0 minimization is an NP-hard problem. One of the methods to solve (5) is to approximate ℓ_0 minimization formulation to ℓ_1 minimization formulation:

$$\widehat{u} = min\ \|u\|_1 \quad s.t. \quad \|\widehat{y} - \Theta u\| < \epsilon. \tag{6}$$

Under the condition of Restricted Isometry Property (RIP) [7], the ℓ_1 problem is provably equivalent to minimizing ℓ_0 problem. The ℓ_1 minimization is convex and can be solved within the polynomial time. Therefore, the reconstructed signal, $\widehat{x}$, is retrieved by:

$$\widehat{x} = \Psi \widehat{u}. \tag{7}$$

3. ADAPTIVE COMPRESSED SENSING ARCHITECTURE

In this section, we present our proposed adaptive CS architecture. First, we introduce the entire framework of adaptive CS. Then we discuss the core component, dynamic knob, in the architecture. Specifically, we analyze the knob design in-depth and model the design criterion in a close form. Finally, we formulate the design optimization into a dynamic programming problem for the optimal design.

3.1 Architecture Overview

Towards efficient sensing in EEG-based BCI, we propose an adaptive CS architecture, which can dynamically reconfigure its own components based on the input EEG signal. The entire architecture in BCI is shown in Figure 1. It consists of four key components, i.e., a dynamic knob module, a randomized encoding module, a quantization module in adaptive CS front-end, and a signal reconstruction module in data aggregator.

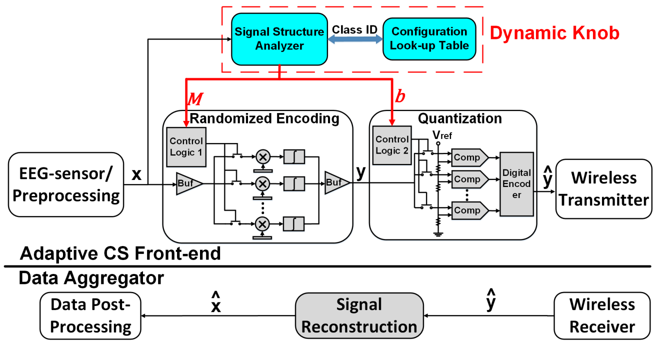

Figure 1: The block diagram of adaptive CS architecture in Wireless BCI.

In Figure 1, original analog signals, which usually denote the raw analog data, $x \in R^N$, coming from sensors, are analyzed by a signal structure analyzer in our proposed dynamic knob. The dynamic knob outputs the optimal parameter estimation to configure the two functional modules: randomized encoding and quantization. Then the analog data x is encoded into an M-dimensional vector, $y \in R^M$, by linear encoding $\Theta_{M \times N}$. Through the quantization scheme $Q_b(.)$, every measurement becomes a certain b-bit digital representation, $\widehat{y}$. A wireless transmitter streams these measurement data to the data aggregator. When the wireless receiver obtains the data extracted from bit stream, the data aggregator performs reconstruction algorithms to recover N-dimension original input signal x from the quantized M-dimension compressed measurement y. We can see that the dynamic knob is the core part in adaptive CS architecture, which acts as the "brain" to control other modules to accommodate the adaptivity of EEG signals.

Here, we introduce the models of energy and performance [8] to evaluate the adaptive CS architecture. In the front-end, power consumption is dominated by the volume of data stream in wireless communication. The energy model can be formulated as follows:

$$E = C \times M \times b, \tag{8}$$

where M is the sampling rate, b is the bit resolution, and C is the wireless communication energy per bit, which is determined by the wireless communication protocol and usually a constant. In addition, we use the percentage root-mean-square difference (PRD) as the performance metric of the entire architecture:

$$PRD = \frac{\|x - \widehat{x}\|_2}{\|x\|_2} \times 100\%, \tag{9}$$

where $\widehat{x}$ denotes the recovered signal and x is the original input signal.

3.2 Dynamic Knob in CS

The dynamic knob framework is the most significant part specifically designed for sensing adaptivity, which needs a low-power and high-accuracy design for the mobility of EEG front-end. The knob is made up of two components, a signal structure analyzer and a configuration look-up table. The look-up table will store the pre-calculated configuration parameters. It can be implemented by the ultra-low non-volatile power memory technology. Therefore, the main challenge in adaptive CS architecture is to design a low-

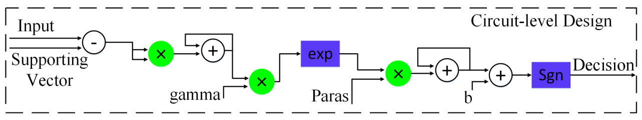

Figure 2: The data flow of SVM testing phase on circuit level.

Table 1: Description of seven attributes in SVM classification node.

Attributes	Description
Parent_Category (*P_c*)	The input category set of this classifier
LChild_Category (*LC_c*)	The subset of *Parent_Category* when the decision of classifier is -1
RChild_Category (*RC_c*)	The subset of *Parent_Category* when the decision of classifier is 1
LChild_domain (*LC_d*)	The SVM node ID set that connects to *LChild_Category*
RChild_domain (*RC_d*)	The SVM node ID set that connects to *RChild_Category*
energy (*eny*)	The average energy consumption to identify an input feature vector
accuracy (*acc*)	Recognition rate to identify training data by the trained SVM classifier

power and high-accuracy signal structure analyzer.

In this part, we introduce a cascade SVM scheme for our signal structure analyzer, whose preliminary element is the binary SVM classifier [9]. We take two factors, accuracy and energy consumption, into consideration simultaneously. The accuracy can be calculated by recognizing the training data using the corresponding classifier. For the energy consumption, we need circuit-level implementation of the basic binary SVM classifier to estimate its power consumption.

We first illustrate the circuit-level implementation of binary SVM. The main task of dynamic knob is to deal with the multi-class classification problem, and we only need to consider the testing phase of SVM. We use a time-division multiplexing strategy to design our implementation, and take radical basis function (RBF) [10] as the kernel in the SVM classifier. Considering that it is computationally complicated to calculate the exponential value on the hardware platform, we adopt the Cordic algorithm [11] to accomplish this task. The circuit-level details of binary SVM implementation is illustrated in Figure 2.

When the input vector functions, we calculate the substraction values between the input vector and supporting vectors, respectively. Then a square operation is executed by the multiplier for individual components of the subtraction vector. Subsequent addition operations sum all the squared values together to a scalar value and prepares for the multiplication with parameter γ. Then the data goes through the exponential calculation module (exp) which applies the Cordic algorithm. Finally, the sum result of the multiplication between exponential value and *paras* can be taken as the indicator for the final binary decision. We can obtain the *paras* as the following:

$$paras_i = y_i \times \alpha_i. \tag{10}$$

Based on the above circuit-level design, we can estimate the energy consumption of each trained binary SVM classifier.

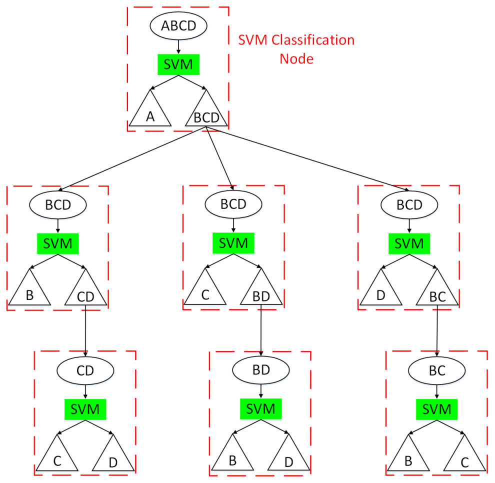

Figure 3: SVM classification tree: tree-shaped organization of the SVM classification nodes.

3.3 Problem Formulation

In this part, we mathematically formulate the design of the signal structure analyzer. We have a large group of trained binary SVM classifiers, and each binary SVM has two key attributes, classification accuracy and energy consumption. Then the most important is to find the optimal SVM-cascade classifier from the complete solution space, i.e., all kinds of possible combinations of elementary binary SVM candidates.

This problem is intractable, if we brute-force search for the optimal solution in the complete solution space. Here, we develop a tree-based method to transform this problem to a solvable form within polynomial time. First, some basic definitions are essential for understanding our problem:

Definition 1. *An SVM classification node is a binary SVM classifier with seven attributes, Parent_Category, LChild_Category, RChild_Category, LChild_domain, RChild_domain, accuracy and energy.*

In Definition 1, we emphasis that each SVM classification node has two child domains (left and right), and each child domain can arbitrarily connect to other SVM nodes, as the red-dashed rectangle shown in Figure 3. Another emphasis is that the attribute *energy* indicates the average energy consumption for a single classification event. For simplicity of presentation, we directly use nJ, instead of nJ/event, as SVM node's energy unit. The detailed attribution description is illustrated in TABLE 1.

Definition 2. *An SVM classification tree is a tree consisting of the SVM classification nodes, with its leaf node indivisible. The connectivity between two nodes is built only when the Parent_Category attribute in child node is equal to the LChild_Category or RChild_Category attribute in parent node.*

According to Definition 2, we can see that an SVM classification tree divides the data set until it cannot be divided, with only one category left. We provide an example of building a 4-class SVM classification tree in Figure 3. There are four classes in the figure, referred to as A, B, C and D. Each red-dashed rectangle indicates an SVM classification node. In the SVM classification node, the top ellipse,

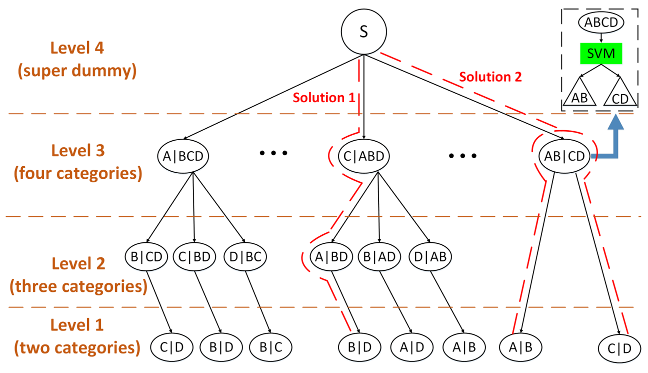

Figure 4: The complete SVM classification tree.

Parent_Category, is the data needed to be classified, and the green rectangle is the SVM classifier. After the decision of the SVM classifier, we obtain two category subsets, *LChild_Category* and *RChild_Category*. For example, the top SVM node is designed to classify the A category from all other three classes, thus its *Parent_Category* is $ABCD$, *LChild_Category* is A and *RChild_Category* is BCD. Since *LChild_Category* is indivisible, with only the A category, we continue with classification of the right child set. Here, we have three choices of how to classify *RChild_Category* into two groups, B and CD, C and BD, D and BC, respectively. We connect all these three classifiers to the right child domain of the previous top node. So the *LChild_domain* attribute of the top node is $\emptyset$, and *RChild_domain* includes all the three nodes dividing the BCD set. There are still three groups with two categories needed to be divided further. We continue to train the subsequent classifiers until there is no node to be split. Therefore, we finished an SVM classification tree build-up.

DEFINITION 3. *A complete SVM classification tree is a tree with a super dummy node connecting all the possible SVM classification trees for a specific multi-class classification problem.*

Based on Definition 3, we build a complete SVM classification tree, including all the possible SVM classification nodes as its tree node, as illustrated in Figure 4. In this complete tree, we define a new delimiter "|" to simplify the symbol of our SVM classification node. This is a binary relation symbol, with the variables before the delimiter classified as the left child and the elements after the delimiter identified as the right child. The union of both children is the entire input data set. For example, we show a detailed SVM classifier for the $AB|CD$ nodes as the dashed black rectangle in Figure 4. Therefore, the design of the signal analyzer changes to how we search for the optimal SVM cascade classifier on the complete SVM classification tree.

3.4 Problem Solution

Before we discuss the solution of this design optimization, we first have a look at some properties of the complete SVM classification tree.

LEMMA 1. *If an SVM classification node subset from the complete SVM classification tree can form a tree with super dummy node as its root, whose leaves are indivisible and with all the classes no replication, this SVM node subset provides a solution for the multi-class classification problem.*

According to Lemma 1, we can find some cascade classifier solutions for the 4-class problem in Figure 4, such as the cascade of $C|ABD$, $A|BD$, $B|D$ and the combination of $AB|CD$, $A|B$, $C|D$. So the solution will be a single path from root to specific leaf, or the multi-path combination, as the $AB|CD$ case, which needs multiple paths to construct the solution. We can see from the above lemma that the subset of classifiers, locating on a single path from the root node to leaf node, is the minimal structural unit to construct the solution for the multi-class classification problem.

LEMMA 2. *The complete SVM classification tree is a complete solution space for the multi-class classification problem.*

From Lemma 2, we can see that our complete SVM tree is the complete solution space of all the possible solutions of multi-class classification. In addition to the Lemma 1, with paths from root to leaf node, we can conclude that the optimal cascade classifier must be a single path or multiple path combination on the complete SVM classification tree.

Our purpose is to find the optimal cascade classifier to compromise accuracy and energy consumption. We define an accuracy density, Ad, as the criterion to evaluate the relationship between accuracy and energy consumption:

$$Ad = \frac{accuracy}{energy}. \tag{11}$$

From the definition of Eq. (11), we can see that Ad is actually an indicator of how much accuracy gain can be obtained by consuming 1 unit of energy for an event. Thus, big Ad means that more accuracy can be reached by consuming the same energy. Therefore, if we have the complete solution set Qs, our objective is to maximize the Ad sum for the cascade SVM classifiers Cp from Qs, as the following:

$$Cp = arg \max_{Cp \in Qs} (\sum_{i \in Cp} Ad_i). \tag{12}$$

Our strategy is not directly searching for the specific traversing path. On the contrary, we divide the complete SVM classification tree into several levels according to the category number of nodes' *Parent_category* attribute, as shown in Figure 4. Then the original design problem can be re-formulated as the multi-stage decision problem. We define $dp[i][j]$ as the max sum of Ad at the j-th node on the i-th level. We apply a dynamic programming algorithm [12] on this complete SVM classification tree to obtain the optimal solution, with its recursive formula as the followings:

$$dp[i][j] = Ad(i,j) + \underbrace{\max_{LC\_d} dp[l_1][k_1]}_{Left\ Child} + \underbrace{\max_{RC\_d} dp[l_2][k_2]}_{Right\ Child}, \tag{13}$$

From the formulation in Eq. (13), we can see that the programming path is proceeding by the connectivity among nodes, not trying to find a specific path. Each node updates its accuracy density accumulation by the sum of the max Ad of left child, the max Ad of right child and the Ad in its own node. The time complexity of Dynamic Programming on our complete SVM tree is $O(n^2)$, where n is the total number of all the SVM classification nodes. After updating all the accuracy density accumulation of SVM nodes, we can obtain the largest Ad, whose programming path is the optimal SVM cascade classifier solution.

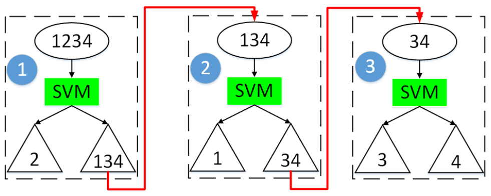

Figure 5: An example of the optimal cascade SVM classifier, consisting of three binary SVM nodes.

Table 2: Node characterization in the optimal solution

ID	*P_c*	*LC_c*	*RC_c*	*acc (%)*	*eny (nJ)*	*Ad (%/nJ)*
1	1234	2	134	86.67	62.45	**1.39**
2	134	1	34	100	76.67	**1.30**
3	34	3	4	100	48.76	**2.05**
Total	–	–	–	–	187.88	**4.74**

4. EXPERIMENTS

In this section, we evaluate our proposed adaptive CS architecture. We characterize the design of dynamic knob first, and then compare the performance between traditional CS and adaptive CS architecture.

4.1 Dynamic Knob Characterization

In this part, we discuss the characterization of our dynamic knob design. We choose the EEG dataset from Physionet [5] as our test bench, initially 120 continuous 128-sample EEG segments with 60 as the training set and the other 60 as the testing set. We take the Libsvm tool [13] to train the basic SVM classifier. For the energy simulation, we implement our dynamic knob design by verilog using verilog compile simulator (VCS) in Synopsys. We use design compiler (DC) to synthesize our design with TSMC 130nm standard cell library and report the power consumption by the Power Compiler.

After obtaining the accuracy and energy attributes, we build the complete SVM tree, including 7 SVM classification trees, for the 4-class classification problem of EEG signals. We execute dynamic programming on the complete SVM classification tree, and the programming path corresponding to the largest *Ad* sum value is the optimal cascade classifier combination. The final optimal solution includes 3 SVM nodes, with the highest *Ad* sum value 4.74 %/nJ. Related node characteristics and their relationship are illustrated in TABLE 2 and Figure 5, respectively.

The optimal solution is a sequential cascade SVMs, as shown in Figure 5. The input signal vector first goes through the number 1 SVM node. If the decision is Category 2, the recognition is terminated, labeling the input signal as Category 2. On the alternative decision, the input feature vector continues to be verified by number 2 SVM node, as the red arrow line shows. If the result is Category 1, the classification task is over. Otherwise, the input vector needs to be identified by the last SVM node, number 3. Based on its decision, the input feature can be recognized as Category 3 or Category 4. The total energy consumption in this analyzing procedure is 187.88 nJ/event, as listed in TABLE 2.

4.2 Adaptive CS v.s. Traditional CS

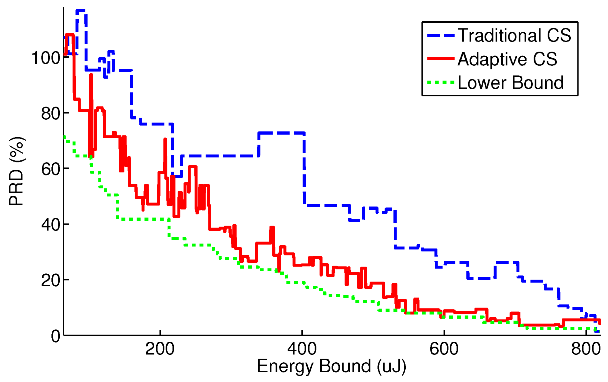

Figure 6: PRD of Segment 55 **under three cases, the traditional CS, the adaptive CS and the lower bound.**

After characterizing the dynamic knob, we would like to examine the PRD improvement in our adaptive CS architecture. We simulate the CS architecture using Matlab on PC with Intel Core i7, 3.4GHz and 8G RAM. We employ inverse discrete wavelet transform (IDWT) as the sparsity-inducing transformation basis, Ψ. All of our experiments use Bernoulli random variables as sensing array and select the uniform quantization strategy.

Towards practical BCI, we adopt an IPv6-based communication model over Bluetooth Low Energy (BLE) on real devices [14] to model energy consumption of wireless data transmission. The throughput of EEG is 0.5 KBytes/s, and the 128-length segment can be transmitted in one packet. Thus, according to the experiment of connection mode of BLE, the energy consumption of this setup is 325 KBytes/J, that is, $C = 0.4$ uJ/bit in the energy model.

For the sake of comparison, we calculate the PRD of both adaptive CS and traditional CS with all energy bounds between 64 and 819 uJ. In the traditional CS architecture, its bit resolution is set as 16 with sampling rate ranges from 10 to 128. The adaptive CS can reconfigure sampling rate M, 10 $\sim$ 128, and bit resolution b, 1 $\sim$ 16. We use all the 60 EEG segments in the testing set to go through the two architectures. We also show the lower bound of PRD from brute-forcing, which is the optimal solution in the entire performance-energy space. Figure 6 illustrates the three PRD curves under each energy bound for the EEG segment 55.

From Figure 6, we can see that the traditional CS, blue dashed line, is always with the largest PRD in all cases of energy bounds. Our adaptive CS, red line, is much better than the traditional constant bit resolution case. As energy bound increases, the PRD of adaptive CS case gradually approximates to the lower bound. this is because larger bit resolution can reduce the truncated data noise from the quantization procedure. Then sampling rate M gradually takes advantage to affect the reconstruction. This indicates that our adaptive CS architecture can adapt to the structure variation in the EEG signal for adaptive sensing.

The average reconstruction quality improvement of all the 60 testing segments are evaluated. For simplicity of data representation, we define $APRD$, the average PRD, as the performance metric on all the energy bounds for a specific

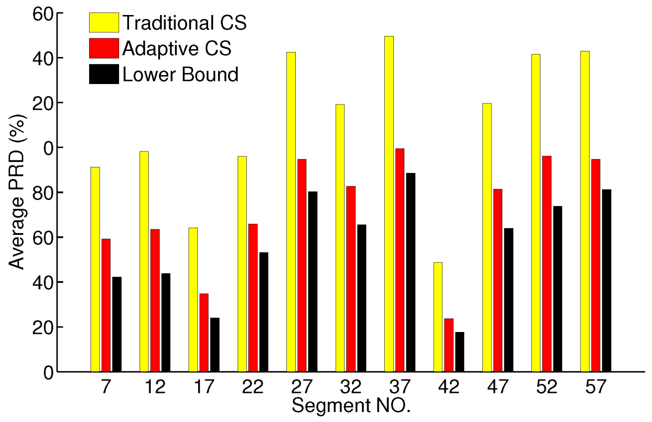

Figure 7: Average PRD of EEG segments under three cases: the traditional CS, the adaptive CS and the lower bound case.

testing segment, as follows:

$$APRD_i = \frac{1}{|EB|}\sum_{j\in EB} PRD(i,j), \tag{14}$$

where EB is the energy bound set, $|EB|$ is the total number of the bounds and $PRD(i,j)$ refers to the reconstruction error under energy bound j of segment i. Based on the Eq. (14), we can plot select average PRD values from the EEG testing set in Figure 7.

In Figure 7, we can see that our adaptive CS, denoted by the red bar, can significantly improve the PRD compared to traditional CS architecture, shown as the yellow bar. On the other hand, the difference between adaptive CS and the lower bound, indicated by the black bar, is small, which indicates that our adaptive CS is close to the optimal performance. Also, we can observe the change of signal intrinsic information over different EEG segments through their PRDs in the lower bound case.

Specifically, we define the relative enhancement index (REI) to quantitatively evaluate the improvement in our adaptive CS architecture, whose form is as follows:

$$REI = \frac{APRD_{TCS} - APRD_{ACS}}{APRD_{TCS} - APRD_{LB}}, \tag{15}$$

where $APRD_{TCS}$ is the average PRD of traditional CS, then $APRD_{ACS}$ and $APRD_{LB}$ refer to the PRD from the adaptive CS case and the lower bound case. Because the average PRD has its lower bound, the improvement should be a ratio by the difference between $APRD_{TCS}$ and $APRD_{LB}$ as the denominator, which is the maximum value in theory, and the improvement contributed by our adaptive CS, $APRD_{TCS} - APRD_{ACS}$, as the numerator. Once calculating the REI for each EEG segment, we can obtain 73.49% relative enhancement of average PRD, resulting from our adaptive CS. Therefore, our adaptive CS architecture can excellently accommodate to EEG signal dynamics in BCI, with its 73.49% reconstructed signal quality improvement.

5. CONCLUSION AND FUTURE WORK

In this paper, we introduced an adaptive CS architecture for BCI. We presented the entire framework of adaptive C-S architecture. Then we proposed a specific dynamic knob design to dynamically control other components. We model the optimal knob design to a dynamic programming formulation. Eventually, we verified our proposed adaptive CS architecture by experiments on continuous EEG signals. We analyzed the characterization of the dynamic knob and discuss its impact on CS architecture. We demonstrated that our adaptive CS architecture can greatly improve the signal reconstruction quality to adapt to signal dynamics in BCI.

In the future work, we will consider designing energy-aware supervised learning methods. Another direction is to apply more accurate supervised learning algorithms to improve the architecture performance towards signal dynamics.

6. ACKNOWLEDGMENTS

This work is in part supported by NSF CNS-1423061/1422417, NSF/AFOSR ECCS-1462498/1462473 and the Ohio Occupational Safety and Health Research Program.

7. REFERENCES

[1] C-T Lin, L-W Ko, M-H Chang, J-R Duann, J-Y Chen, T-P Su, and T-P Jung. Review of wireless and wearable electroencephalogram systems and brain-computer interfaces–a mini-review. *Gerontology*, 56(1):112–119, 2009.

[2] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[3] Matthew L Malloy and Robert D Nowak. Near-optimal adaptive compressed sensing. In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pages 1935–1939. IEEE, 2012.

[4] Xing Wang, Wenbin Guo, Yang Lu, and Wenbo Wang. Adaptive compressive sampling for wideband signals. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–5. IEEE, 2011.

[5] George B Moody, Roger G Mark, and Ary L Goldberger. Physionet: a web-based resource for the study of physiologic signals. *IEEE Eng Med Biol Mag*, 20(3):70–75, 2001.

[6] Aosen Wang, Wenyao Xu, Zhanpeng Jin, and Fang Gong. Quantization effects in an analog-to-information front-end in eeg tele-monitoring. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 62(2):104–108, 2014.

[7] Argyrios Zymnis, Stephen Boyd, and Emmanuel Candes. Compressed sensing with quantized measurements. *Signal Processing Letters, IEEE*, 17(2):149–152, 2010.

[8] Aosen Wang, Chen Song, and Wenyao Xu. A configurable quantized compressed sensing architecture for low-power tele-monitoring. In *Green Computing Conference (IGCC), 2014 International*, pages 1–10. IEEE, 2014.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[10] S Łukaszyk. A new concept of probability metric and its applications in approximation of scattered data sets. *Computational Mechanics*, 33(4):299–304, 2004.

[11] Jack E Volder. The cordic trigonometric computing technique. *Electronic Computers, IRE Transactions on*, (3):330–334, 1959.

[12] Eric V Denardo. *Dynamic programming: models and applications*. Courier Dover Publications, 2003.

[13] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[14] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 232–237. IEEE, 2012.