ERROR-ADAPTIVE CLASSIFIER BOOSTING (EACB): EXPLOITING DATA-DRIVEN TRAINING FOR HIGHLY FAULT-TOLERANT HARDWARE

Zhuo Wang[†] *Robert Schapire*[⋆] *Naveen Verma*[†]

[†] Department of Electrical Engineering, Princeton University, Princeton, NJ, USA 08544
[⋆]Department of Computer Science, Princeton University, Princeton, NJ, USA 08544

ABSTRACT

Technological scaling and system-complexity scaling have dramatically increased the prevalence of hardware faults, to the point where traditional approaches based on design margining are becoming inviable. The challenges are exacerbated in embedded sensing applications due to the severe energy constraints. Given the importance of classification functions in such applications, this paper presents an architecture for overcoming faults within a classification processor. The approach exploits machine learning for modeling not only complex sensor signals but also error manifestations due to hardware faults. Adaptive boosting is exploited in the architecture for performing iterative data-driven training. This is used to enable the effects of faults in preceding iterations to be modeled and overcome during subsequent iterations. We demonstrate an entire system integrating the proposed classifier, capable of training its model entirely within the architecture by generating estimated training labels. FPGA experiments show that high fault rates (affecting $>$3% of all circuit nodes) occurring on $>$80% of the hardware can be overcome, restoring system performance to fault-free levels.

Index Terms— Boosting, Circuit faults, Fault tolerance, Machine learning, Sensor systems

1. INTRODUCTION

Machine-learning algorithms are becoming increasingly important in embedded sensing applications. Machine learning enables efficient construction of *data-driven* models for analyzing signals that are too complex to otherwise model analytically. Given the prominence of recognition/detection functions [1], frameworks for classification and regression are of particular importance. Recently, studies have also begun to expose the potential that machine learning brings for overcoming non-idealities (technological faults, transistor variations, etc.) affecting the hardware platform itself. In [2], an approach called data-driven hardware resilience (DDHR) is presented that enables very high levels of fault tolerance by utilizing a machine-learning stage to model the variances in embedded data caused not only due to the application signals but also due to hardware faults. However, in DDHR the machine-learning stage is explicitly required to be fault protected. Two system demonstrations showed that by protecting $7\% \mid 30\%$ of the hardware, performance essentially equivalent to a fault-free system could thus be achieved even with faults affecting $0.02\% \mid 3\%$ of the circuit nodes in the rest of the architecture (resulting in bit error rates of 20-50%). The problem is that the complexity of machine-learning kernels scales strongly with the models required [3], making their impact on a system substantial, particularly as the hardware platform and application signals both scale in complexity.

The aim of this work is to create an architecture for the machine-learning stage (classifier) that is *itself* allowed to be greatly affected by faults. The presented approach, termed error-adaptive classifier boosting (EACB), takes advantage of adaptive boosting (AdaBoost) [4], which is an iterative training algorithm applied to weak classifiers. The contributions of this work are as follows: (1) we present a classifier architecture and training algorithm wherein high-levels of randomized hardware faults are overcome via iterative data-driven training over weak classifiers; (2) we present a specialized circuit implementation, and its various design points, to enable 80-90% of the classifier hardware to be tolerant against faults, dramatically reducing the fault-protected hardware required in a DDHR system; (3) we demonstrate a system in hardware via an FPGA platform, which permits controlled characterization, showing that performance is consistently restored even with randomized faults affecting 3% of the classifier circuit nodes.

Support is provided by SRC, NSF (CCF-1253670), and Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

2. BACKGROUND

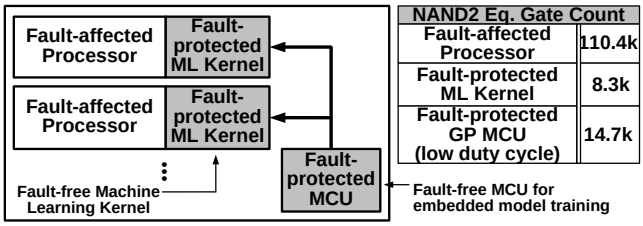
Below, DDHR is introduced to illustrate the powerful opportunities that machine learning enables for overcoming hardware faults via data-driven training. Then AdaBoost is introduced, which we will exploit in a machine-learning kernel that is itself allowed to be highly fault prone, substantially expanding the concept of DDHR.

2.1. Data Driven Hardware Resilience (DDHR)

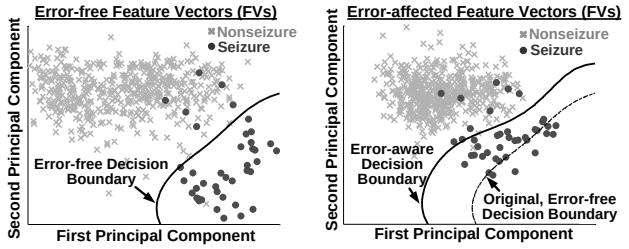
The key to DDHR [2] is utilizing data from an instance of fault-affected hardware to construct a model for classification or regression. The resulting model is called an *error-aware model*; while, generally, faults occur randomly and cause unpredictable errors, the error-aware model represents the data statistics *in the presence of the particular occurring faults*. Fig. 1 shows a DDHR system for embedded sensing. The fault-affected blocks (in white) include feature-extraction processors. The fault-protected blocks (in grey) include a support-vector machine (SVM) classifier [5] and a microcontroller for applying and training the model, respectively. Training, however, requires labels in addition to error-affected data. The labels are achieved entirely within the architecture by implementing a temporary error-free system on the microcontroller, which can thus employ a generic model not requiring training to particular error statistics; since training is performed infrequently, the temporary system has minimal impact on the system. While the labels, thus computed, are *estimates* rather than *ground truths*, they enable model training that converges to give performance up to that of a fault-free system. Fig. 1a shows the amount of fault-protected and fault-affected

hardware derived from RTL synthesis of a demonstration system for EEG-based seizure detection [6].

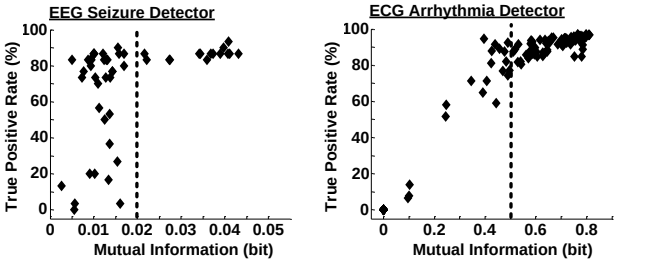
Fig. 1b shows the error-aware model that results in the case of actual faults (from FPGA emulation). An important characteristic of DDHR is that, thanks to data-driven training, system performance is not limited by the rate or magnitude of errors, but rather more fundamentally by the level of *information* retained in the error-affected data; Fig. 1c shows the correspondence exhibited between performance and mutual information for two demonstrated DDHR systems [6]. A primary limitation of DDHR, however, is the need for substantial fault-protected hardware (machine-learning stages), whose impact increases with increasing system and application-data complexity due to the need for higher-order models [3]. Accordingly, we aim to extend the error-modeling capabilities within the classifier hardware itself. For this we leverage the AdaBoost algorithm, which uses multiple weak classifiers. We show that these enable an architecture wherein high-levels of faults can be overcome through iterative training.



(a) DDHR architecture leverages fault-protected blocks.



(b) Model trains to faults (data from seizure-detection system).



(c) DDHR perfromance corresponds with mutual information.

Fig. 1. DDHR overcomes faults through training in fault-protected machine-learning stages.

2.2. Adaptive Boosting (AdaBoost)

AdaBoost is a machine-learning algorithm that aims to achieve a highly-accurate classifier through a combination of T weak classifiers [4]. The output of a weak classifier is (arbitrarily) weakly correlated with the true class. The algorithm iteratively trains the weak classifiers, establishing both a decision rule and a weight for each iteration. The final hypothesis is then derived from weighted voting over the weak classifiers. So long as each weak classifier performs slightly better than random guessing, the performance is guaranteed to fit a training set perfectly, given enough iterations. However, an important consideration that remains is choosing a weak classifier that results in good generalization over testing data. A common and effective weak classifier used with AdaBoost is the decision tree [7] (training details described in [8]). Each node of the tree is a statement about the feature vector being classified, thus determining the next node to be considered, eventually yielding a classification result at the leaf nodes.

In practical weak-classifier implementations, as in the case of a decision tree, performance is typically limited by an inadequate decision rule for fitting the training data [4]. In this work, the concept is extended, with weak-classifier performance also being limited by *errors due to hardware faults*. We focus on decision trees, not only because they are empirically shown to be effective weak classifiers, but also because they can be mapped to an implementation that substantially mitigates the amount of control circuitry, thereby minimizing the fault-protected hardware required. Additionally, decision trees bring the benefit of comparatively simple training algorithms. Nonetheless, training remains substantially more complex than real-time classification. To perform training at run time (with limited data), we develop an algorithm that leverages the idea of the FilterBoost algorithm [9], but while also substantially reducing the computations and embedded memory required. This thus minimizes the overhead of an embedded trainer.

3. ERROR-ADAPTIVE CLASSIFIER BOOSTING

The aims of EACB are as follows: (1) strong classification, with minimal hardware energy and complexity, based on scalable data-driven training; (2) high classifier performance in the presence of very high fault rates; (3) need for minimal fault-protected hardware, both for classification and training. The following subsections describe the EACB architecture and implementation.

3.1. EACB Architecture

EACB is based on the following recognition. A stage whose output function is determined by data-driven training over its set of inputs raises the possibility of overcoming faults in the preceding stages. The errors from faults in the preceding stages can be viewed simply as altering the statistics of the resulting data. EACB uses AdaBoost, wherein the hypotheses generated by preceding weak classifiers are taken as inputs during data-driven training of subsequent iterations. The architecture of EACB is shown in Fig. 2, consisting of the following: (1) T fault-affected weak classifiers, implemented as decision trees; (2) a fault-protected voter, implemented as a T-input signed adder where the inputs and sign bits correspond to the classifier weights and outputs, respectively; and (3) a fault-protected trainer, which is required infrequently and is implemented via a microcontroller. Using AdaBoost, the weak classifiers effectively enable data-driven training in successive stages. Consequently, each iteration performs training to the statistics of the hypotheses generated by the preceding weak classifiers *in the presence of their faults*. However, as in the case of DDHR, training the weak classifiers requires training labels. A temporary, fault-free classifier is thus implemented in software on the microcontroller to generate estimated labels (as in DDHR); we will show (Sec. 4) that this is effective for restoring performance to the level of a fault-free classifier. The gate counts from RTL synthesis of a system (described in Sec. 4) are provided for the various blocks, showing that the architecture is achieved with minimal fault-protected hardware.

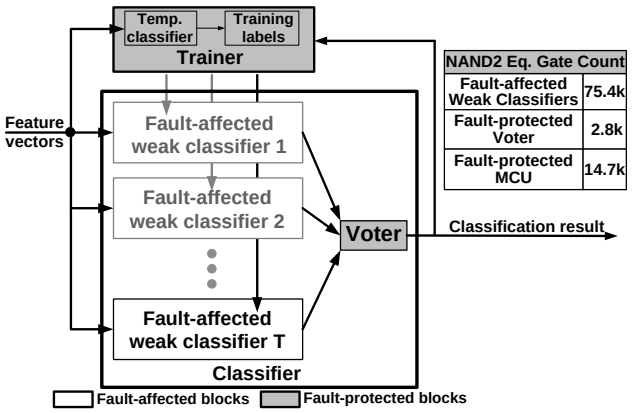


Fig. 2. Architecture of EACB.

3.2. Weak Classifiers for Maximizing Fault Tolerance

The choice and implementation of the weak classifiers strongly influences overall performance (i.e., tradeoff between accuracy and diversity [10]), training complexity, and achievable level of fault tolerance. Among the various classifiers that have been considered for boosting (support vector machines [11], neural networks [12], decision trees [7][10]), decision trees enable reduced training complexity and, as described below, enable a specialized implementation that offers high fault tolerance within EACB.

A critical aspect for fault tolerance is a circuit's control-path implementation. While data-path faults alter the output statistics, the probability of retaining some correlation with class information remains high, as required of weak learners in AdaBoost. However, control-path faults can result in degenerate outputs, inadequate for even a weak classifier. Fig. 3 shows the implementation developed for the decision-tree weak classifiers, with the aim of minimizing the control path while retaining the programmability needed for EACB training. The implementation consists of three stages. First, a node for each of the n features is implemented by digital comparison (CMP) with a threshold derived from model training; this has the benefit of immediately reducing the n features to n bits, corresponding to the node outputs. Second, m n-to-1 multiplexers (MUX) select the nodes and their locations to include in the tree, as also derived from model training. The number of nodes is thus limited to m. Third, the m multiplexer outputs are used as the index to a look-up table (LUT), whose entries are also determined from training, thereby deriving the single-bit classifier output. In this implementation, only the number of tree nodes is limited; analysis in Sec. 4 considers the impact on EACB of various sized trees.

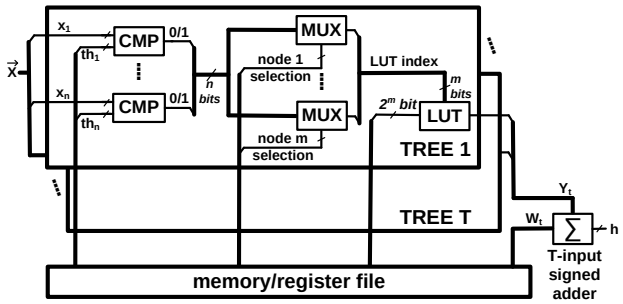


Fig. 3. Implementation of m-node decision-tree.

3.3. Low-overhead Embedded Trainer

The challenge with embedded training is the need for a large training set (to address diversity), thus making memory requirements excessive. For example, a standard training algorithm [4] in the system of Sec. 4 would require 5k feature vectors, corresponding to 420kB of memory. We develop a training algorithm that reduces the training-data memory through two approaches: (1) feature selection based on a learner metric; and (2) iterative training with small but distinct training sets to mitigate generalization error. For feature selection, each feature is ranked based on its number of occurrences in the decision trees formed during an off-line training phase (i.e., for the temporary classifier of Fig. 2); the most commonly occurring features are selected as being the most informative for classification. For enabling small, distinct training sets, the idea of FilterBoost is leveraged [9], wherein new training data is selected for each iteration. However, for run-time training, where the only data available is being acquired on line, we use all the acquired data to form the training set. This in fact is critical for reducing computational complexity, by avoiding the need to derive complex selection criteria, thereby reducing the number of clock cycles of the microcontroller by a factor of $10\times$. Results (Sec. 4) show that the approach gives convergent performance with a standard training algorithm while reducing the memory required by over a factor of $50\times$.

4. EXPERIMENTS

To evaluate EACB, we perform hardware experiments using an FPGA. This permits error injection at desired rates and in a randomized manner, enabling controlled characterization. The experimentation details are provided below.

4.1. Application Demonstration and Design Space

For experimental demonstration and evaluation, we apply EACB to a system for EEG-based detection of epileptic seizures. The system consists of a feature-extraction stage and a classifier (which employs EACB). The features correspond to the spectral-energy distribution of 2 EEG channels, across 7 frequency bins, over three 2-second epochs, giving a total of 42 features [13]. The classifier consists of the architecture in Fig. 2, with the trainer implemented via an embedded OpenMSP microcontroller [14] running software for training and label estimation. EEG data for testing (10k seconds) is obtained from the MIT-CHB seizure database [15].

The decision-tree weak classifiers are implemented in RTL, using the topology in Fig. 3. As noted, the maximum number of nodes in the tree is set by the topology. For design exploration, we consider three cases: (1) 7-node trees; (2) 4-node trees; and (3) 1-node trees (i.e., stumps). The metrics for evaluation include the following: (1) the fault-rates tolerable while maintaining application-level performance; (2) the amount of fault-affected and fault-protected hardware; and (3) the complexity of the trainer.

4.2. Experimental Approach

The experimentation flow, based on FPGA emulation, is shown in Fig. 4. The demonstration system is designed in Verilog RTL and synthesized into a gate-level netlist using an ASIC logic library. Faults are then introduced within the gate-level netlist. The fault model we focus on is a stuck-at-1/0 fault, which is representative of a wide range of physical faults and is appropriate for representing limiting failures in low-energy (low-voltage) operating modes [16]. To introduce the faults, the gate-level netlist is edited (via a script) by including (1) multiplexers on a large number of nodes (set by FPGA

mapping limits), and (2) a fault-control module within the system. The resulting netlist is then mapped to the FPGA. Each multiplexer drives the corresponding node with either the intended signal from the synthesized netlist or with a static signal fixed to logic 1/0. Both the multiplexer select signal and the static input signal come from the fault-control module. The fault-control module consists of a register file that can thus programmably configure instances of faults to be activated. Recognizing that, in addition to the rate of faults, the actual nodes affected can have a strong impact on errors, multiple (five) randomized fault instantiations are considered for each fault rate. A second FPGA is used strictly as an Ethernet transceiver, to load configuration data into the fault-control module and to send/receive data from the system to a host PC.

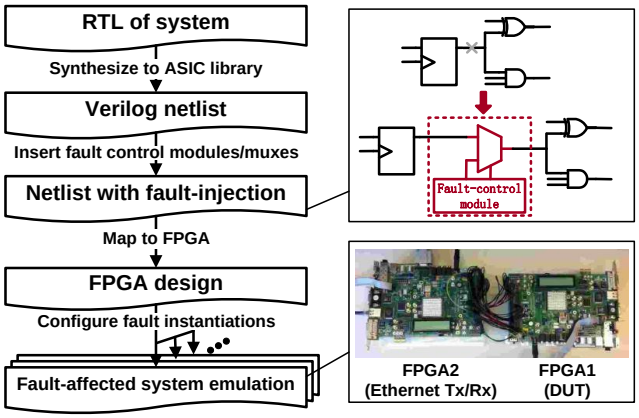


Fig. 4. FPGA-based flow for experimentation.

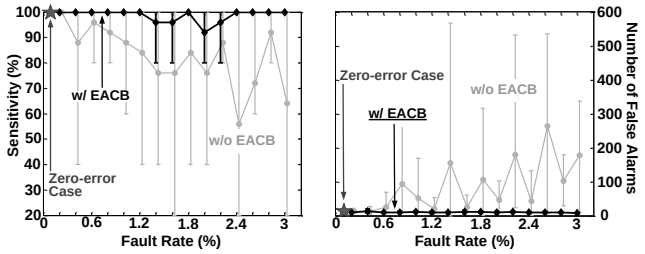
4.3. Results

Below, the measured results following design synthesis (to the ASIC netlist) and FPGA testing are provided for the evaluation metrics.

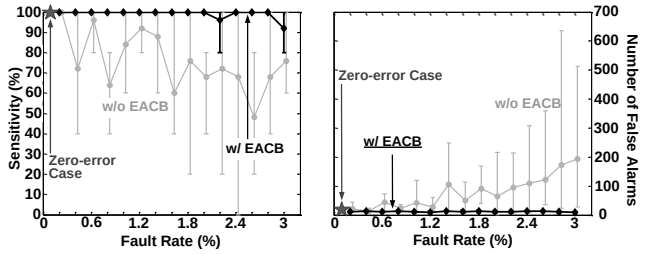
Fault tolerance. For the three decision-tree topologies considered, faults are introduced on the circuit nodes at a rate from 0% (representing fault-free system performance) to 3%; higher fault rates are limited by FPGA mapping constraints. Five cases of fault instantiations are considered at each rate. The measured performance is shown in Fig. 5, both with EACB and without it (i.e., the usual case wherein classifier training is performed offline using a standard algorithm with the ideal weak classifiers). While the performance without EACB degrades rapidly, with EACB the performance is consistently and substantially restored for all decision-tree topologies.

Need for fault-protection. The ratio of fault-protected hardware needed is impacted by the decision-tree topology in two ways: (1) it changes the balance of hardware required for the weak classifiers versus the voter; and (2) it changes the number of iterations T required for achieving the application-level performance. Fig. 6 shows the number of iterations and the total gate counts for the three cases. In all cases, well over 80% of the classifier hardware can be affected by faults while maintaining performance. Though larger trees appear to fare somewhat better in terms of both fault-affected hardware and total hardware (gate count), the trainer complexity (below) favors smaller trees.

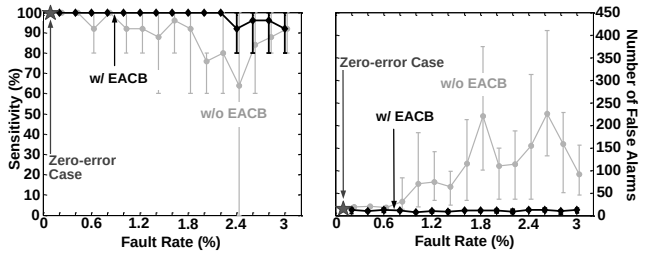
Trainer complexity. Though larger trees result in reduced classifier hardware, the challenge is that they require substantially more training data during each iteration. Otherwise, they suffer substantial overfitting. This strongly influences the training complexity and the amount of embedded memory required within the trainer. As shown in Fig. 6, stumps require just 7.2kB of memory and 0.33G microcontroller clock cycles for training over all iterations, using the algorithm from Sec. 3.3. 7-node trees require 13.6kB and 2.6G clock-cycles, respectively.



(a) system performance with stumps classifiers



(b) system performance with 4-node trees classifiers



(c) system performance with 7-node trees classifiers

Fig. 5. Performance with and without EACB (error bars show worst/best performance over five fault-injected instantiations) to 3% fault rates (higher rates limited by FPGA mapping constraints).

Weak Learner Type	Stump	4 nodes	7 nodes
Fault Rate Tolerable	> 3%	> 3%	> 3%
# Iterations T for Convergence	64	22	19
Equivalent NAND Gates per Tree	1169	1250	1575
Total Hardware Gates	74815	27493	28997
Fault-affected Hardware	82.0%	86.5%	89.4%
Trainer memory (kB)	7.2	10.4	13.6
Trainer clock cycles ($x10^9$)	0.33	2.8	2.6

Fig. 6. Comparison of EACB systems for three weak classifier topologies.

5. CONCLUSIONS

This paper presents the approach of error-adaptive classifier boosting (EACB), which employs iterative data-driven training to construct a classifier that is trained to the statistics generated by errors due to its own hardware faults. An architecture based on EACB is developed that maximizes fault tolerance, through an implementation based on reduced control path, and that minimizes trainer complexity, through a modified FilterBoost algorithm. Hardware experiments using an FPGA demonstrate the ability to overcome faults affecting 3% of the circuit nodes, on >80% of the architecture, implying high fault tolerance and the need for minimal fault-protected hardware.

6. REFERENCES

[1] P. Dubey, "Recognition, mining and synthesis moves computers to the era of tera," *Tech. at Intel Magazine*, 2005.

[2] N. Verma, K. H. Lee, K. J. Jang, and A. Shoeb, "Enabling system-level platform resilience through embedded data-driven inference capabilities in electronic devices," in *IEEE Int. Conference on Acoustics, Speech and Signal Processing*, Mar. 2012.

[3] K. H. Lee and N. Verma, "A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 7, pp. 1625–1637, 2013.

[4] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*, The MIT Press, 2012.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel Based Learning Methods*, Cambridge University Press, 2000.

[6] Z. Wang, K. H. Lee, and N Verma, "Overcoming computational errors in low-power sensing platforms through embedded machine-learning kernels," *in review IEEE Tran. on VLSI*.

[7] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *International Conference on Machine Learning*, 2006, pp. 161–168.

[8] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[9] J. K. Bradley and R. Schapire, "FilterBoost: Regression and classification on large datasets," in *Advances in Neural Information Processing Systems*, J.C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., pp. 185–192. MIT Press, Cambridge, MA, 2008.

[10] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, Aug. 2000.

[11] X. Li, L. Wang, and E. Sung, "Adaboost with svm-based component classifiers," *Engineering Applications of Arti?cial Intelligence*, vol. 21, no. 5, pp. 785–795, Aug. 2008.

[12] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural Computation*, vol. 12, no. 8, pp. 1869–1887, Aug. 2000.

[13] A. Shoeb and J. Guttag, "Application of machine learning to epileptic seizure detection," in *International Conference on Machine Learning*, June 2010.

[14] O. Girard, "openmsp430," .

[15] PhysioNet, "CHB-MIT scalp EEG database," .

[16] N. Verma, J. Kwong, and A. P. Ch, "Nanometer mosfet variation in minimum energy subthreshold circuits," *IEEE Transactions on Electron Devices*, vol. 55, no. 1, pp. 163–174, Jan. 2008.