Genetic Programming for Energy-Efficient and Energy-Scalable Approximate Feature Computation in Embedded Inference Systems

Jie Lu, *Student Member, IEEE*, Hongyang Jia, *Student Member, IEEE*, Naveen Verma, *Member, IEEE*, and Niraj K. Jha, *Fellow, IEEE*

Abstract—With the increasing interest in deploying embedded sensors in a range of applications, there is also interest in deploying embedded inference capabilities. Doing so under the strict and often variable energy constraints of the embedded platforms requires algorithmic, in addition to circuit and architectural, approaches to reducing energy. A broad approach that has recently received considerable attention in the context of inference systems is approximate computing. This stems from the observation that many inference systems exhibit various forms of tolerance to data noise. While some systems have demonstrated significant approximation-versus-energy knobs to exploit this, they have been applicable to specific kernels and architectures; the more generally available knobs have been relatively weak, resulting in large data noise for relatively modest energy savings (e.g., voltage overscaling, bit-precision scaling). In this work, we explore the use of genetic programming (GP) to compute approximate features. Further, we leverage a method that enhances tolerance to feature-data noise through directed retraining of the inference stage. Previous work in GP has shown that it generalizes well to enable approximation of a broad range of computations, raising the potential for broad applicability of the proposed approach. The focus on feature extraction is deliberate because they involve diverse, often highly nonlinear, operations, challenging general applicability of energy-reducing approaches. We evaluate the proposed methodologies through two case studies, based on energy modeling of a custom low-power microprocessor with a classification accelerator. The first case study is on electroencephalogram-based seizure detection. We find that the choice of two primitive functions (square root, subtraction) out of seven possible primitive functions (addition, subtraction, multiplication, logarithm, exponential, square root, and square) enables us to approximate features in 0.41mJ per feature vector (FV), as compared to 4.79mJ per FV required for baseline feature extraction. This represents a feature extraction energy reduction of 11.68$\times$. The important system-level performance metrics for seizure detection are sensitivity, latency, and number of false alarms per hour. Our set of GP models achieves 100 percent sensitivity, 4.37 second latency, and 0.15 false alarms per hour. The baseline performance is 100 percent sensitivity, 3.84 second latency, and 0.06 false alarms per hour. The second case study is on electrocardiogram-based arrhythmia detection. In this case, just one primitive function (multiplication) suffices to approximate features in 1.13μJ per FV, as compared to 11.69μJ per FV required for baseline feature extraction. This represents a feature extraction energy reduction of 10.35$\times$. The important system-level metrics in this case are sensitivity, specificity, and accuracy. Our set of GP models achieves 81.17 percent sensitivity, 80.63 percent specificity, and 81.86 percent accuracy, whereas the baseline achieves 82.05 percent sensitivity, 88.12 percent specificity, and 87.92 percent accuracy. These case studies demonstrate the possibility of a significant reduction in feature extraction energy at the expense of a slight degradation in system performance.

Index Terms—Approximate computing, energy efficiency, error-aware inference, feature extraction, genetic programming, machine learning

1 Introduction

Machine learning (ML) is being widely used to discern patterns from sensor signals that are too complex to model analytically. Since such pattern recognition is of importance in diverse domains, ML systems have had a broad impact. The focus of our work is on energy-constrained sensing applications, where energy-intensive signal processing can quickly drain the battery.

Traditional benefits of technology scaling enable significant energy savings. However, the effects are diminishing with the slowing of Moore's Law. As a result, architectural innovations, such as heterogeneous and multi-core processors, have become crucial to achieving performance and energy gains [1], [2]. Approximation has added an additional exploitable dimension, namely accuracy of computational results [3]. Specifically, approximate computing exploits the gap between the acceptable level of accuracy in an application and the accuracy of a traditional computing system. With the extra knob, we can trade computation accuracy for both energy and performance.

Although the core principles of approximate computing are not new and have been exploited in many fields, recent efforts have been targeted at implementing approximate computing in all layers of the computing system, from circuits to architectures and software/algorithms. For example,

• *Authors are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544.*
E-mail: {jielu, hjia, nverma, jha}@princeton.edu.

Manuscript received 27 Mar. 2017; revised 13 July 2017; accepted 6 Aug. 2017. Date of publication 10 Aug. 2017; date of current version 22 Jan. 2018.
(Corresponding author: Jie Lu.)
Recommended for acceptance by E. Kursun.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2017.2738642

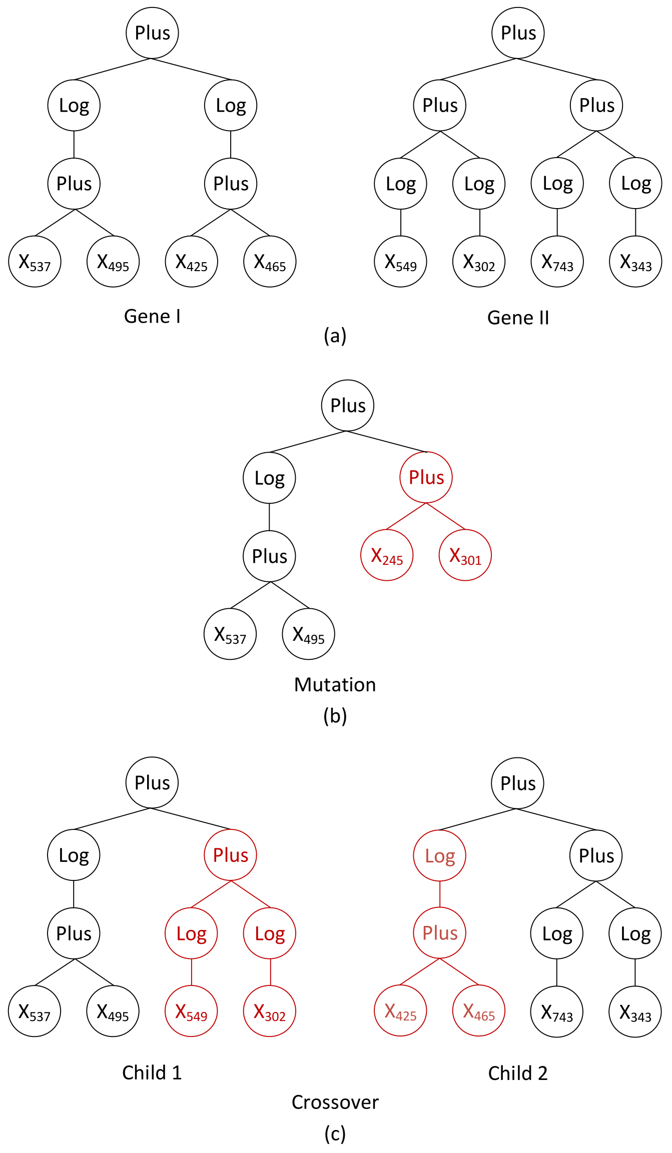


Fig. 1. (a) Example of two genes from two individuals: Gene I implements the function $log(X_{537}+X_{495})+log(X_{425}+X_{465})$ and Gene II implements the function $log(X_{549})+log(X_{302})+log(X_{743})+log(X_{343})$, (b) mutation of Gene I, and (c) crossover between Genes I and II: the red sections are the modified parts.

hardware-related efforts include voltage scaling [4] at the circuit level and precision scaling [4], [5] at the microarchitecture level. Software-related efforts include memoization [6].

We propose an algorithmic approximate computing approach where the intermediate results of an ML system are approximated to reduce the computation energy. Specifically, we approximate computations in the traditionally energy-intensive feature-extraction (FE) stage of an ML system with the aim of achieving significant energy efficiency. The proposed work makes synergistic use of two concepts:

- Genetic programming (GP): GP is a probabilistic search algorithm that iteratively optimizes a set of objectives utilizing principles of natural selection [7] to approximate nonlinear computations via a small set of primitive functions (PFs). An example of such a set is (addition, subtraction, multiplication, logarithm, exponential, square root, and square).
- Error-aware inference: The principle employed for error-aware inference stems from the concept of data-driven hardware resilience (DDHR) [8], [9]. DDHR has previously been used to adapt inference models to specifically overcome the impact of non-ideal hardware, which causes computational errors, to perform accurate inference.

Though GP was originally targeted at computer program design, it has been successfully applied to many other fields, including performance improvement of ML systems [10], [11]. It is capable of searching high-dimensional spaces to approximate nonlinear computations with relative ease. Importantly, it allows us to constrain the search process in specific ways, which we leverage to enable energy scalability in the resulting GP model.

Traditional GP techniques utilize a tree structure comprising PFs and terminals. These functions and terminals are chosen at random when constructing the computation tree. In doing so, we can arrive at a solution without prior assumptions. In this paper, we use a multigene variation of GP [12]. Each member of the GP population is a weighted linear combination of the outputs from a set of GP trees. Each tree represents a "gene". Thus, genes can be easily converted to free-form arithmetic equations. Fig. 1a shows two genes used to approximate two of the features from a 768-sample epoch of one electroencephalogram (EEG) channel. Both genes contain two types of PFs: logarithm and addition. X_i corresponds to a particular sample of the EEG channel.

Through the structure of a GP model, many flexible mappings from inputs to outputs are possible, enabling approximation of a diverse range of nonlinear functions. Generally, the approximation error reduces if more complex models are used, as we will demonstrate later. Specifically, by complexity we refer to the number of genes and the depth of genes. For example, in Fig. 1a, both Genes I and II have the same model complexity with a maximum tree depth of four and number of genes per model of one. Generally, less complex GP models, which may exhibit larger approximation error, have lower energy consumption. This gives rise to the energy-versus-approximation knob for FE that we exploit. However, in inference systems, our interest is not in approximation of features, but rather ultimately in the accuracy of inference (e.g., classification). Therefore, we enhance this knob by substantially recovering from feature-approximation errors using the DDHR concept, which was previously developed to overcome hardware-level non-idealities (such as digital-logic faults) [8], [9]. This enables system performance to be maintained with relatively simple GP models for FE, thus enabling FE energy to be reduced with relatively modest degradation in system performance. This approach has also recently been employed in a custom microprocessor, which focuses on an accelerator for executing GP models [13]. In this work, we provide a detailed analysis of the approximation methodology and demonstrate its efficacy via energy modeling based on a custom low-power general-purpose microprocessor [implemented in 130 nm complementary metal-oxide semiconductor (CMOS), operating at 1.2V], wherein the GP models corresponding to FE are implemented on a central processing unit (CPU) and the inference computations are implemented on a configurable classification accelerator [14].

The objectives of this work are as follows:

1. Using GP implemented on a CPU to approximate features in the FE stage of an ML system, thereby reducing the number of operations and energy required.
2. Using the concept of DDHR to learn an inference model in the presence of the GP-approximation errors, thereby enhancing the accuracy of inference results as approximation errors increase.

The remainder of the paper is organized as follows. Section 2 presents background material on approximate computing, GP, error-aware inference models, and related work. Section 3 gives details of our system. Section 4 presents measurement results for various applications operating on real data. Section 5 presents concluding remarks.

2 Background

In this section, we give a brief overview of approximate computing and the two concepts we utilize to achieve approximate computing: GP and error-aware inference. We also discuss related works on GP and ML systems.

2.1 Approximate Computing

Approximate computing takes advantage of error tolerance in applications and achieves energy gains at the cost of result accuracy [15]. The concept of approximate computing can be traced to several well-established disciplines, e.g., compression algorithms for images, audio, and video, and networking protocols for packet delivery [16]. In the past few years, approximate computing has been used at device, architecture, and software/algorithm levels in order to improve computing system performance or to decrease overall system energy [15]. Current approximate computing methods include, but are not limited to, the following.

Voltage scaling trades off accuracy with energy consumption. For example, voltage overscaling is used where the supply voltage is aggressively lowered below nominal voltage, which can reduce energy drastically. This energy reduction is achieved at the possible expense of timing violations, which can introduce error in the final result [15].

Precision scaling reduces the required storage or computing power consumption by changing the bit-width or the precision of operands [4], [5].

Inexact hardware refers to the use of inexact circuits. For example, Kahng and Kang have presented an inexact adder [17]. It utilizes sub-adders that operate on different segments of an N-bit summation. The sub-adders are independent of each other so that carry propagation is redued in order to reduce critical-path delay. As the segment length is increased, fewer sub-adders are used, and the chance of getting the correct result increases. However, fewer sub-adders also lead to an increase in dynamic power consumption (due to increased critical path delay leading to increased voltage requirement).

Memoization works by storing the results of functions to be fetched when later an identical function or input is encountered [15]. In approximate computing, instead of using an exact match in terms of function or inputs, approximate value reuse reduces the number of required operations [6].

Load value approximation is useful when a load miss occurs in a cache. Instead of fetching from the next-level cache or main memory, which would induce large latency, this approximation estimates the load values. In doing so, the processor no longer needs to stall for a fetch. This reduces the number of memory accesses [18].

Approximate neural networks take advantage of neural network learning capabilities and their resilience to structural changes [15]. Zhang et al. have proposed a method in which a neuron is defined as resilient if a small perturbation of its computation leads to a minimal change in the result [5]. The most resilient neurons are approximated via precision scaling, memory access skipping or inexact hardware to reduce overall computation energy.

Algorithmic approaches take two general directions in approximate computing. The first direction is to utilize algorithms for more efficient implementations of the approximate computing techniques mentioned above, e.g., dynamic bitwidth adaptation and error-correction algorithms [19]. The second direction is to reduce the energy of the baseline algorithms by reducing the number of calculations. For example, Chippa et al. found that the number of support vectors in support-vector machines (SVMs) correlates well with the quality of SVMs and the algorithm's energy consumption [20]. Thus, an approximate output can be computed by ordering the support vectors in terms of their importance and utilizing only the most important ones for inference. Esmaelizadeh et al., on the other hand, proposed multilayer perceptrons to approximate sections of algorithms, such as fast Fourier transform (FFT), clustering, and image processing [3].

Somewhat differently from the above-mentioned approaches to approximate computing, we focus on approximating the whole process of feature extraction and any computations it involves. This gives rise to new knobs for scaling energy, trading overall feature value accuracy for energy efficiency.

2.2 Genetic Programming

GP is a method for evolving solutions to transform a set of mathematical objects or functions into a new set of objects or functions using the Darwinian principle of reproduction and survival of the fittest [7]. It has been deployed in a wide range of applications, e.g., pattern recognition in images, speech recognition, and classification systems [11], [21], [22]. It has the following common features:

1) *Algorithm derivation based on a population of individuals:* Typically, an individual consists of a set of mathematical objects or functions, called genes. A gene is represented by a computation tree whose nodes are PFs and leaves are either variables or constants. The first generation of individuals is generated randomly. Thereafter, each generation of individuals evolves from the previous generation.
2) *Genetic material diversification based on inheritance:* The population evolves with the help of genetic operators. The main genetic operators are reproduction, crossover, and mutation. Reproduction involves exactly copying individuals from the current generation into the next generation based on fitness value ranking of the individuals in the population. Crossover swaps a part of the genetic material between two individuals, whereas mutation randomly

changes a portion of the genetic material of one individual. The individuals are selected based on fitness value ranking to participate in crossover or mutation. Therefore, individuals with high fitness value rankings are more likely to participate in building the next generation. Fig. 1b shows the results of a mutation of Gene I and Fig. 1c shows the children resulted from crossover operation between Gene I and Gene II shown in Fig. 1a.

3) *Model selection:* In each generation, model selection is based on the fitness value of each individual. The fitness value is calculated using a defined fitness function. As an example, the fitness function may be based on the difference between the GP model-produced output value and the ground truth. In this case, a lower fitness value indicates higher quality. Thus, such an individual has a higher probability of being selected for breeding the next generation. Hence, genetic materials with lower fitness values are more likely to survive in the evolutionary process.

As mentioned before, GP genes are usually realized as computation trees. The leaves are either variables or constants, and are referred to as terminal nodes. The branch nodes correspond to operators or functions, and are called nonterminal nodes. The set of allowed operators and functions is called the PF set. The set of all variables and constants is called the terminal set. Two conditions must be satisfied before GP can be successfully applied to a specific problem [7]: sufficiency and closure. Sufficiency indicates that the terminal and nonterminal sets together must be capable of representing a solution to the problem [7]. Closure requires that each PF should be applicable to all values of the input [7]. For example, logarithm of value 0 would produce an invalid output and thus does not meet the closure criterion. This can be addressed by using a protected logarithm that yields a 0 in this case to satisfy the closure criterion.

2.3 Error-Aware Inference

DDHR [8], [9] was originally used to perform inference in the presence of hardware faults. It utilizes data-driven training to construct a model for inference from error-affected data. The trained inference model is then able to classify newly observed error-affected data. This newly trained classification model is called the *error-aware inference model*. There is no restriction on the type of classification model that is suitable for error-aware inference. It should just be able to sufficiently correct the errors in error-affected data. The error source is assumed to be static, i.e., both the training data and any future test data are assumed to be affected by it, thus assumed to have the same underlying statistics [9].

An important characteristic of DDHR is that, due to data-driven training, the system performance is limited by the level of information retained in the error-affected data. For instance, in [23], the empirical mutual information between error-affected feature data and class membership is shown to set the level of classification performance achievable by the error-aware model. Thus, even when the approximation error is large, as long as the mutual information is preserved, classification accuracy does not degrade. We examine the relationship between approximation error and classification accuracy in Section 4.

In this paper, we use the error-aware inference concept in another domain: to provide resilience against errors arising from the use of GP to approximate the outputs of nonlinear signal processing in the FE stage. The approximation error source is static in this case, since the same set of GP models approximates the feature vectors used by the inference system.

2.4 Related Work

To improve the energy efficiency of ML systems, accelerators tailored to a specific kernel of the ML system and application are often employed. For example, accelerators for efficient arithmetic or FFT implementations, which can be used in the FE stage, have been presented [24], [25]. Energy-efficient accelerators for classifiers [26], [27], [28] have also been presented. However, these approaches are limited in the computations they can address. We tackle this shortcoming by focusing on a flexible platform targeted at arbitrary ML applications, where diverse FE computations are implemented on a CPU and inference computations, performed through a specific kernel, are implemented on a fixed-function SVM accelerator. Further, the FE computations are based on GP models, to which our approximation methodology can be applied. Recent work has demonstrated a platform wherein GP models are also implemented on an accelerator, yielding significant energy reduction but somewhat restricting the model structure, e.g., choice of PFs [13].

In the context of GP, most studies have focused on the projection of input vectors to new feature spaces or the modeling of classifiers [11], [29], [30], [31], [32], [33], [34]. One area of research in GP-related FE focuses on using GP to automatically construct and select features [29], [30], [31]. Instead of using predefined features as the baseline for fitness functions, these studies use the fitness function based on the number of classification errors. In doing so, the training phase needs a tremendous number of classifiers to be trained and used. Specifically, the number of classifiers required is equal to the product of GP population size and number of generations.

Unlike the GP-based FE approaches discussed above, we do not propose to construct new features or manipulate FE inputs. Specifically, we are not proposing FE or feature selection methods that utilize inference systems as part of fitness evaluation. Instead, we aim to develop a methodology to approximate baseline features (these are features extracted by the application-specific FE method) with a very low operation count. In doing so, we can avoid possible dependencies on the classifier choices and classifier parameters. Thus, if for a set of features, different types of classifiers are used, we do not need to adjust the GP models accordingly. This baseline feature approximation method enables us to replace the original FE method with GP models based on a small set of PFs.

GP has also been extensively studied as a classification framework [11], [32], [33], [34], [35], [36]. Multiple GP models are used in an ensemble to evaluate class-imbalanced data or multiclass data [32], [33]. GP models have been integrated with other types of classifiers as well, the most studied of which is the neural net [34], [35], [36].

In this paper, we do not target GP-driven classifiers. The reason is that the approximation error of the GP-driven classifier cannot be corrected since classification is the last stage

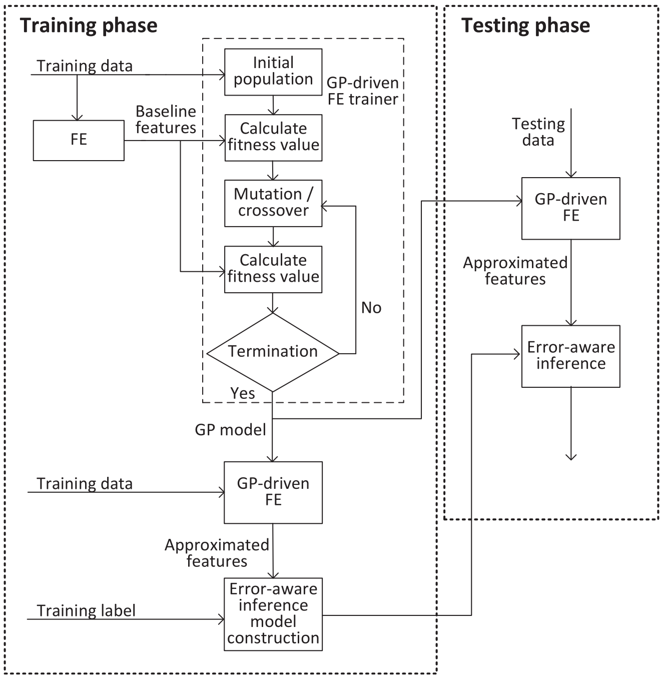


Fig. 2. Design flow of training phase and testing phase.

of an ML system. Thus, when GP models are used to implement the classifier, they need to have minimal approximation error to maximize classification accuracy. This requires significantly increased GP model complexity that, unfortunately, does not yield much energy efficiency over traditional classifiers, e.g., SVM.

3 System Details

In this section, we describe how GP can be used to approximate FE. We also discuss system implementation and provide system energy measurements.

3.1 System Formulation

The features used in ML systems vary drastically across applications [37]. Thanks to GP's ability to dynamically adapt its structures to the problem environment [7], we can use it to tackle different feature types and develop a versatile energy-efficient ML system.

The design flow for the proposed system is shown in Fig. 2. Fig. 2 shows the flow for the training and testing phases, which include the construction of GP-driven FE and error-aware inference model. During the training phase, an initial population is built using the pool of PFs shown in Table 1. These PFs are targeted due to the availability of energy-efficient implementations for them and their ability to approximate nonlinear functions. Note that additional functions can be employed in GP, such as distribution-altering functions (e.g., normalization, sorting) and order-dependent functions (e.g., FFT) [31]. These functions, however, require vector inputs and thus violate the closure requirement. Therefore, we do not incorporate them as part of the PF set.

The GP models are built with a set number of genes and a specified maximum gene depth. A total of M GP models are constructed for an M-dimensional FV, i.e., one GP model per feature. A gene within a GP model is realized in the form of a computation tree, and gene depth represents the depth of the tree. A trade-off exists between the number of genes and gene depth for obtaining a better feature quality, which we discuss later. An individual produces a set of results based on input values. Specifically, each gene within the individual produces a result. One possibility is to accumulate the gene results using multivariate regression

$$y'_j = \alpha_j + \beta_{j1} Gene_{j1} + \beta_{j2} Gene_{j2} + \cdots + \beta_{jk} Gene_{jk}, \tag{1}$$

where y'_j is the GP-approximated jth feature in an M-dimensional FV and β_{jk} is the weight assigned to gene output $Gene_{jk}$, i.e., the kth gene output of the jth feature. The GP-induced output, y'_j, is compared with the baseline feature (i.e., the feature extracted by the application-specific FE method) to determine the fitness of an individual model. After the population of models has evolved through a given number of generations, the best model can be picked based on fitness values.

The fitness function evaluates the quality (e.g., output accuracy) of GP models so that the best GP model receives the best score. In this sense, the fitness function guides the evolution to find a model that optimizes the desired qualities (e.g., output accuracy). In our flow, we use the following fitness function

$$f_j = \sqrt{\frac{\sum_{i=1}^{N} (y_{i,j} - y'_{i,j})^2}{N}}, \tag{2}$$

where f_j is the fitness value of the GP model that approximates the jth feature, $y_{i,j}$ represents the jth feature of the ith baseline FV extracted by the original FE method, $y'_{i,j}$ is the GP-approximated jth feature of the ith GP-approximated FV, and N is the number of FVs. Minimizing this fitness function is equivalent to maximizing the feature quality defined in Eq. (3).

$$\text{Feature Quality}_j = (1 - \frac{\sum_{i=1}^{N} (y_{i,j} - y'_{i,j})^2}{\sum_{i=1}^{N} (y_{i,j} - \bar{y}_j)^2}) \times 100\%, \tag{3}$$

where $\bar{y}_j$ is the jth feature averaged over N baseline FVs. If the approximated feature perfectly matches the actual feature, the numerator of the equation becomes 0, producing a

TABLE 1
The PF and Terminal Sets for Evolving GP Models for FE

Symbol	Arity	Description	Symbol	Arity	Description
+,-	2	addition, subtraction	x	0	input data
*	2	multiplication	y	0	feature
log,exp	1	logarithm, exponential	sqrt, square	1	square root, square

Symbols represent operations, functions, and variables. Arity is the number of operands the operation or function accepts.

TABLE 2
Common GP Settings for All Experiments

Population size	500
Selection method	Tournament selection with tournament size of 10
Maximum #generations	1,000
Stopping criterion	Maximum #generations explored or fitness value == 0
Operators	Crossover, mutation
Original population	Ramped half and half (half full-depth tree, half random-depth tree)

maximum value of 100 percent for feature quality. The feature quality of an FV is the average over M feature qualities.

Individuals with lower fitness function values are favored in the GP selection process for participation in creating the next generation of individuals. The next generation is created either through exact copying of an individual, random mutation of an individual, or crossover of genes between two individuals. The specific parameters used are shown in Table 2.

Once the new generation of individuals is created, a new round of fitness function evaluations takes place. If there exists an individual that can perfectly model the outputs or if the population has evolved 1,000 generations, we terminate the evolution process and pick the best individual with the lowest fitness function value. An evolution length of 1,000 generations was found to be adequate. A further increase in the number of generations did not lower the fitness function value. The features, which may incur an approximation error, are then used to construct an error-aware inference model.

In the testing phase, the GP model obtained from the training phase is used for GP-driven FE, as shown in Fig. 2. The system performance is obtained from the error-aware inference model's classification results derived from GP-approximated features. Note that the data used for GP model training/selection are different from the data used for presenting the final performance results.

3.2 System Modeling

To evaluate and compare the energy requirements of the proposed methodology with the baseline approach, we perform energy modeling based on a heterogeneous microprocessor. The microprocessor employs a CPU and a configurable classification accelerator. The GP-driven FE models are assumed to be implemented on the integrated general-purpose CPU. The classification accelerator implements an SVM. SVMs have gained popularity due to their computational efficiency and training robustness to noise and bias. However, the computations required for a complex SVM model may dominate system energy consumption if implemented in software [14], [38]. Thus, an SVM accelerator (SVMA) helps to reduce overall system energy. FE, on the other hand, requires a high degree of programmability as it is strongly tied to the application and its signals. Thus, a CPU is employed for implementing GP-driven FE. The 16b processor architecture that embodies the above requirements is shown in Fig. 3 [14]. It is implemented in 130nm CMOS and operates at 1.2V.

TABLE 3
FE and Classification Energy for Two Applications Mapped to the Processor

Application	FE Energy	Classification Energy	Ratio$*$
Seizure	4.79 mJ per FV	45.10 μJ per classification	106.2 x
Arrhythmia	11.69 μJ per FV	0.24 μJ per classification	48.7 x

$*$*Ratio = (FE Energy) / (Classification Energy)*

Table 3 shows the energy profiling results for two applications discussed further in Section 4, where FE is done via the baseline application-specific FE method. We can see that use of the accelerator dramatically reduces classification energy relative to FE energy. This provides the motivation for reducing the FE energy through the proposed methodology.

Fig. 4 shows the die photo and provides the integrated circuit summary, including the measured energy per cycle for the CPU and SVMA modules [14]. The CPU has a MSP430-compatible instruction set, 16kB of program memory, and 16 kB of data memory. We perform cycle-accurate simulations for this processor through the IAR Systems Embedded Workbench meant for MSP430 [39]. This Workbench is an integrated development environment for C language compilation and MSP430 microprocessor simulation [40], [41]. The simulated number of cycles needed for extracting one FV from the input signals can thus be obtained for each application. We then use the number of cycles and the measured energy per cycle of various processor components (see Fig. 4) to obtain the GP-driven FE energy per FV.

The SVMA executes classification kernels by providing operand and model data to the data-path unit (DPU). It enables programmable partitioning of the local 32kB support vector memory (SV Mem.) to instantiate multiple classifiers with various kernel functions [linear, polynomial, radial-basis function (RBF)] [14]. DPU performs the arithmetic required on FVs for SVMA. A coordinate rotation digital computer (CORDIC) engine is used for hardware-efficient

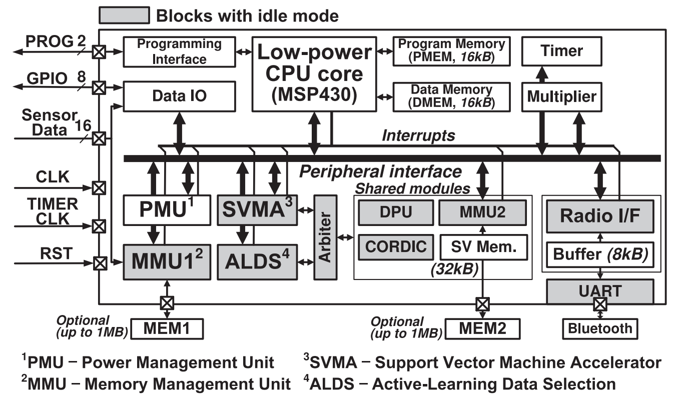
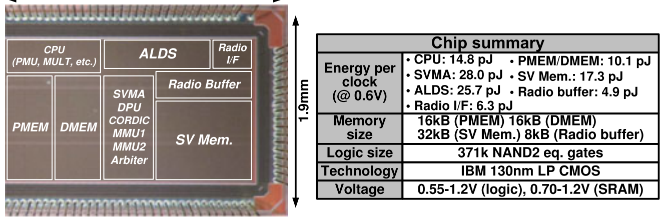


Fig. 3. Processor architecture with ML accelerators [14].



Chip summary	
Energy per clock (@ 0.6V)	• CPU: 14.8 pJ • PMEM/DMEM: 10.1 pJ • SVMA: 28.0 pJ • SV Mem.: 17.3 pJ • ALDS: 25.7 pJ • Radio buffer: 4.9 pJ • Radio I/F: 6.3 pJ
Memory size	16kB (PMEM) 16kB (DMEM) 32kB (SV Mem.) 8kB (Radio buffer)
Logic size	371k NAND2 eq. gates
Technology	IBM 130nm LP CMOS
Voltage	0.55-1.2V (logic), 0.70-1.2V (SRAM)

Fig. 4. Die photo and prototype integrated circuit summary [14].

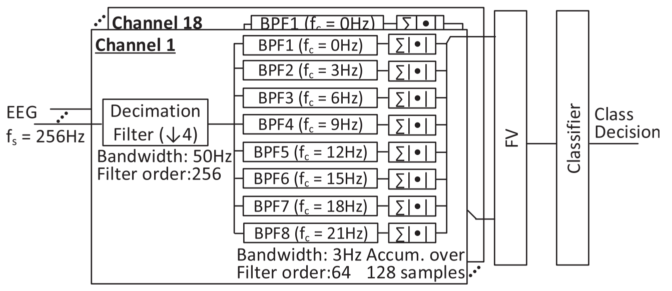


Fig. 5. System diagram for EEG-based seizure detection using baseline features.

computations of the exponential function used in the RBF kernel. The classification energy evaluation method is described in Section 4.3.

In the case-study analysis, we compare system performances in terms of inference results and FE energy to demonstrate the energy-performance tradeoff GP can achieve. Based on the method described above, the FE energy is calculated for all GP models and represents a knob we can tune to obtain the best energy-performance tradeoff. In order to tune this knob, we vary the gene depth and number of genes to limit the maximum number of computations allowed for each GP model. As we tune this energy knob, we demonstrate how close we can keep the inference results to baseline inference results (calculated using baseline features and inferences).

4 Case Studies

We analyze the proposed methodology through two case studies. The first study involves seizure detection based on scalp EEG recordings. The second study involves arrhythmia detection using electrocardiogram (ECG) recordings.

4.1 Case 1: EEG-Based Seizure Detection

For the EEG data, we used recordings from the CHB-MIT database [42]. This database consists of EEG recordings from pediatric subjects with intractable seizures. The recordings are categorized into seizure or non-seizure segments according to expert analysis. All recordings are sampled at 256 Hz. For each second, the EEG signal is measured using 18 channels. Fig. 5 demonstrates the system approach to seizure detection, based on baseline features. For each channel, we segment continuous EEG signals into three-second signal epochs with a two-second overlap per epoch. Each second of EEG data contains 256 samples, resulting in 768 samples per epoch.

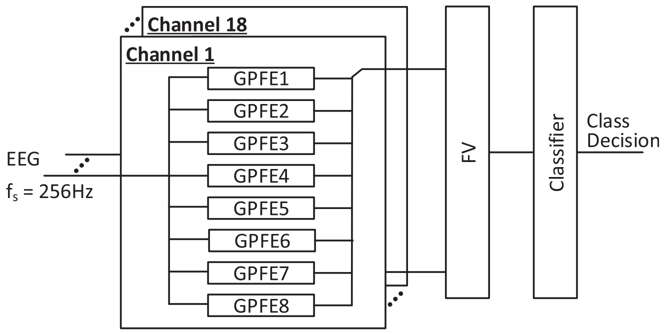
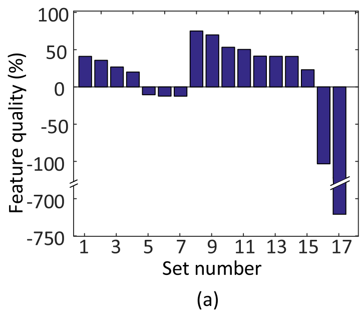


Fig. 6. System diagram for seizure detection with GP-driven FE.



Set number	1 PF	Set number	2 PFs
1	sqrt	8	sqrt,-
2	log	9	log,-
3	*	10	log,+
4	sqr	11	sqrt,*
5	exp	12	sqrt,sqr
6	-	13	log,exp
7	+	14	sqrt,log
		15	exp,*
		16	log,*
		17	log,sqr

(b)

Fig. 7. Feature quality versus PFs used for EEG-based seizure detection: (a) Feature quality over different sets of GP models, and (b) PFs involved in GP model sets 1 to 7 with one PF and sets 8 to 17 with two PFs.

In the baseline FE method, we first decimate the data to 64 Hz. This is equivalent to using 192 samples per epoch. Decimation is necessary to reduce the computational requirements of the band-pass filters (BPFs). The down-sampled samples are then processed through eight BPFs, each with a 3 Hz passband, covering the frequency within a 0-24 Hz range. Only 128 of the filtered samples are used to eliminate the edge effect. The features correspond to the energy of the output samples from each filter over the entire epoch. This requires summing the absolute values of the filter output samples. This represents energy accumulation. Thus, eight BPFs produce eight features per channel. The features from three different epochs are then concatenated to capture transitional variances in the EEG signals. This final FV thus has a dimensionality of $3 \times 8 \times 18$ for the 18 EEG channels.

In our system implementation, the FE stage has three distinct steps: (i) decimation, (ii) linear filtering, and (iii) approximate energy accumulation (which is nonlinear). We model all three steps using GP. Therefore, for the eight BPFs, eight GP models are obtained, as shown in Fig. 6. The final concatenated FV is sent to an RBF-SVM classifier.

We evaluate seizure detection system performance through sensitivity, latency, and number of false alarms per hour. Sensitivity is defined as the fraction of correctly detected seizures over the number of seizures in the entire dataset. Latency is defined as the number of seconds elapsed between the expert-identified seizure onset and system-detected seizure onset. The number of false alarms per hour is defined as the average number of falsely detected seizures per hour in the entire dataset.

4.1.1 GP Model Selection

Fig. 7 shows the feature quality for different PF combinations, all using the same number of genes and gene depth (20 genes, depth 7). All models with just one PF are included. Models based on selected sets of two PFs are also included to demonstrate the wide range of feature qualities achievable. Note that certain models that use two PFs result in lower feature qualities than models with just one PF. This strongly suggests that these models are not the globally optimal solutions for the set of parameters given. To escape from such local optima, one could consider using adaptive probabilities for crossover and mutation [43]. For the purpose of this study, however, we find that relying on the combination of two PFs with the highest feature quality is sufficient for approximating FE in the EEG-based seizure detection application.

TABLE 4
GP Model Results for Seizure Detection Application

Number of genes	Gene depth	GP-driven FE energy (mJ/FV)	Feature quality (%)		Error-aware inference results						Non-error-aware inference results					
					Sensitivity (%)		Latency (sec)		# of false alarms/hr		Sensitivity (%)		Latency (sec)		# of false alarms/hr	
			Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	2	0.02	9.16	0.37	93.33	0.00	24.03	0.07	0.53	0.03	91.67	0.00	21.54	3.07	4.27	0.12
	3	0.03	23.71	0.30	100.00	0.00	7.69	0.35	0.42	0.04	100.00	0.00	13.38	0.87	3.67	0.93
	4	0.05	29.27	0.64	100.00	0.00	7.42	0.20	0.37	0.04	100.00	0.00	8.86	1.04	5.13	0.87
	5	**0.08**	**40.67**	**0.53**	**100.00**	**0.00**	**6.31**	**0.25**	**0.31**	**0.05**	**100.00**	**0.00**	**7.36**	**0.50**	**2.12**	**0.69**
	6	0.15	49.90	1.54	100.00	0.00	5.56	0.17	0.30	0.05	100.00	0.00	6.11	0.53	1.19	0.20
	7	0.22	55.53	0.71	100.00	0.00	4.83	0.16	0.21	0.03	100.00	0.00	5.62	0.65	1.00	0.38
5	2	0.13	30.35	0.07	100.00	0.00	10.33	0.93	1.91	0.41	100.00	0.00	9.66	0.25	3.75	0.20
	3	0.14	53.29	0.23	100.00	0.00	4.87	0.18	0.29	0.04	100.00	0.00	5.20	0.24	0.83	0.15
	4	0.20	56.90	0.35	100.00	0.00	5.01	0.09	0.22	0.02	100.00	0.00	5.17	0.11	0.67	0.18
	5	0.34	60.73	0.34	100.00	0.00	4.90	0.23	0.18	0.02	100.00	0.00	4.75	0.19	0.89	0.17
	6	0.50	64.57	0.42	100.00	0.00	4.56	0.12	0.20	0.05	100.00	0.00	4.49	0.17	0.65	0.08
	7	0.70	66.36	0.48	100.00	0.00	4.49	0.21	0.12	0.03	100.00	0.00	4.64	0.07	0.29	0.15
10	2	0.25	36.35	0.08	100.00	0.00	9.52	0.44	1.82	0.10	100.00	0.00	9.74	0.22	3.71	0.57
	3	0.29	61.37	0.27	100.00	0.00	4.62	0.26	0.20	0.04	100.00	0.00	4.69	0.12	1.20	0.08
	4	0.41	64.29	0.13	100.00	0.00	4.37	0.22	0.15	0.02	100.00	0.00	4.53	0.10	0.43	0.17
	5	0.58	66.97	0.25	100.00	0.00	4.43	0.16	0.12	0.03	100.00	0.00	4.24	0.13	0.73	0.18
	6	0.77	69.07	0.34	100.00	0.00	4.10	0.19	0.12	0.03	100.00	0.00	4.29	0.04	0.43	0.08
	7	1.03	70.72	0.45	100.00	0.00	4.24	0.10	0.10	0.01	100.00	0.00	4.15	0.10	0.50	0.04
20	2	0.49	39.35	0.06	100.00	0.00	9.43	0.16	1.07	0.10	100.00	0.00	9.26	0.26	4.56	0.10
	3	0.58	66.82	0.10	100.00	0.00	4.23	0.09	0.15	0.03	100.00	0.00	4.60	0.21	0.52	0.11
	4	0.83	70.28	0.24	100.00	0.00	4.12	0.08	0.11	0.03	100.00	0.00	4.12	0.11	0.43	0.07
	5	1.03	72.28	0.43	100.00	0.00	4.06	0.10	0.11	0.02	100.00	0.00	3.91	0.11	0.47	0.14
	6	**1.33**	**73.69**	**0.22**	**100.00**	**0.00**	**4.13**	**0.08**	**0.08**	**0.02**	**100.00**	**0.00**	**4.12**	**0.08**	**0.23**	**0.03**
	7	1.59	74.71	0.31	100.00	0.00	3.97	0.11	0.10	0.02	100.00	0.00	4.18	0.12	0.17	0.03

The GP-driven FE energy is presented with the average results and standard deviations of feature quality, error-aware inference results and non-error-aware inference results. Bold results correspond to the high-energy and low-energy models mentioned within the texts.

At most only two PFs are considered because the combination of square root and subtraction has one of the highest feature qualities, while using the least number of computations (i.e., the number of non-terminal nodes). Specifically, the set of GP models for this combination has an average feature quality of 75.09 percent. This feature quality is 2.52 percent better than the average feature quality of the set of GP models built with all seven PFs. The additional PFs add complexity to the models, leading to a slow and difficult search for the optimal solution. Our feature quality optimization method is a greedy search, which means that the models can get stuck in a sub-optimal solution. In this case, the sub-optimal solution with additional PFs does not outperform the models built with fewer PFs. The set of GP models with the highest feature quality based on three PFs (square root, subtraction, addition) improves the average feature quality by less than 1.00 percent while requiring 20 percent more nodes compared with the GP models based on square root and subtraction. These additional nodes are either terminal nodes that require memory access or non-terminal nodes for PF computations. This adds substantially to energy consumption. Hence, when taking into account the above-mentioned trade-off between feature quality and number of nodes, the GP models based on square root and subtraction appear to offer the best choice. This is what we choose for further system analysis.

4.1.2 System Analysis

Next, we evaluate the FE energy required to extract an FV from EEG signals, as shown in Fig. 6. We use GP models based on square root and subtraction, but with a different number of genes and gene depths. The different number of genes are 1, 5, 10, and 20. The gene depths range from 2 to 7. These numbers are chosen to avoid potential over-fitting with more complex GP models [44], [45]. Because of the variations that exist during GP model building, we ran the algorithm with and without error-aware inference five times for all patients and present the average performance with respective standard deviation in Table 4. The average performance of these runs is also shown in Fig. 8. It enables us to analyze the trade-off between FE energy and performance. We base performance evaluation on three criteria: sensitivity, latency, and number of false alarms per hour. The baseline performance is 100 percent sensitivity, 3.84 seconds latency, and 0.06 false alarms per hour at an FE energy of $4.79mJ$ per FV.

Fig. 8a plots feature quality with respect to FE energy. As we scale down the FE energy by decreasing the GP model's number of genes (from 20 genes) or gene depth (from a depth of 7), we see a gradual decrease in feature quality until the FE energy drops below $0.20mJ$ per FV. This suggests the possibility for extensive energy scaling. As we scale down the number of genes and gene depth, we approximate the features with fewer and smaller trees. This

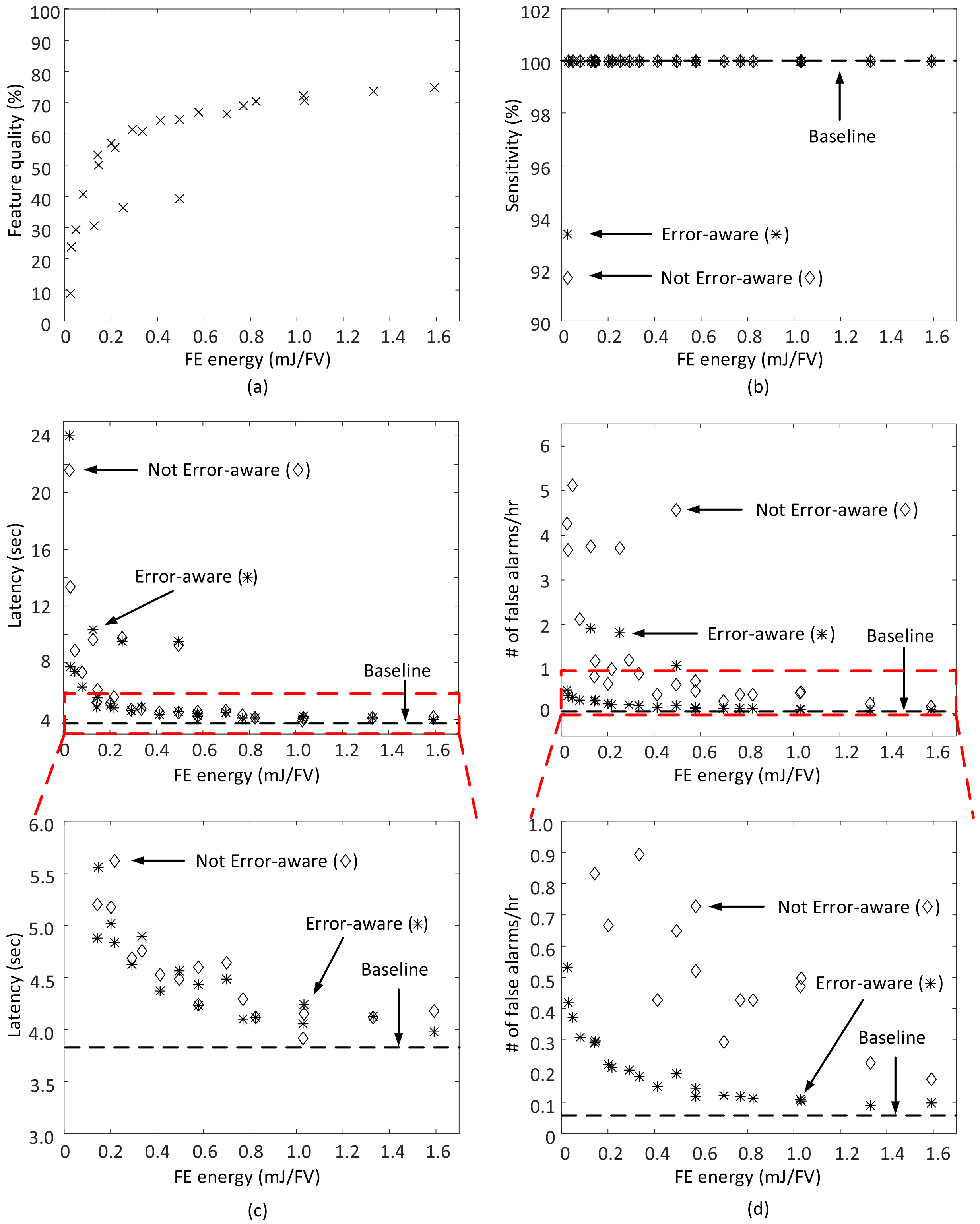


Fig. 8. Seizure detection performance with respect to FE energy: (a) Feature quality versus FE energy, (b) sensitivity versus FE energy, (c) detection latency versus FE energy and a zoomed-in view of the detection latency versus FE energy figure, and (d) number of false alarms per hour versus FE energy and a zoomed-in view of the number of false alarms per hour versus FE energy figure.

reduction in size leads to fewer computations, and thus a decrease in FE energy. At $0.20mJ$ (5 genes, depth of 4), the GP-driven FE energy is reduced from the most energy-intensive GP model by $1.39mJ$ per FV (7.95×) at the cost of 17.81 percent reduction in feature quality. Below $0.20mJ$, the feature quality degrades significantly while only reducing FE energy slightly. For example, the lowest feature quality GP model reduces FE energy from the highest feature quality GP model by $1.57mJ$ per FV (63.75×) at the cost of 65.55 percent reduction in feature quality.

There is, however, a special case where the gene depth is too small to enable construction of an accurate model. For example, the two points, where the feature qualities are between 30 percent and 40 percent and the FE energy is

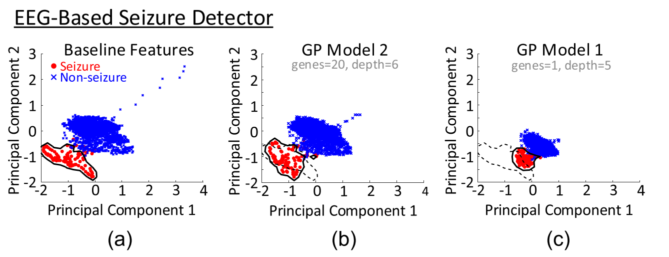


Fig. 9. RBF-SVM classifier boundary for (a) baseline features, (b) GP-approximated features (PFs: square root and subtraction, genes: 20, depth: 6), and (c) GP-approximated features (PFs: square root and subtraction, gene: 1, depth: 5).

between 0.25mJ to 0.49mJ per FV, correspond to GP models with 10 and 20 genes with the same gene depth of 2. At this gene depth, the increase in the number of genes from 10 to 20 results in a 0.24mJ increase in FE energy while only improving the feature quality by 3.00 percent (from 36.35 to 39.35 percent). Meanwhile, if we increase the gene depth by 1 for the GP model with 10 genes and gene depth of 2, we increase the FE energy by 0.04mJ (to 0.29mJ) while improving the feature quality by 36.44 percent (from 36.35 to 61.37 percent). In this case, increasing the gene depth results in greater improvement in feature quality than increasing the number of genes.

Figs. 8b, 8c, and Fig. 8d plot the sensitivity, latency, and number of false alarms per hour, respectively. We show two sets of results for a given set of GP-approximated features. The two sets differ in the inference functions used. One set uses error-aware inference functions, where the GP-induced approximation errors are accounted for during the classification of the test data. The second set uses inference functions trained from the baseline features and thus does not account for the approximation errors. We compare the two sets of results in detail in the next section. Here, we use error-aware inference results to analyze how FE energy scaling using GP models affects performance. The GP models consisting of square root and subtraction with 20 genes and depth 6 perform the best. This set of models requires 1.33mJ per FV, reducing the FE energy by 3.60×. It achieves a sensitivity of 100 percent, latency of 4.13 seconds, and false alarm rate of 0.08 per hour, which compares favorably with baseline performance. As GP-driven FE energy decreases, performance is still maintained with the use of error-aware inference model. Even a GP model that consumes 0.08mJ per FV (40.67 percent feature quality, gene: 1, depth: 5) achieves a sensitivity of 100 percent, latency of 6.31 seconds, and false alarm rate of 0.31 per hour. This reduces FE energy by 59.88× relative to the baseline at the expense of a moderate performance degradation. The GP models with 10 genes and depth 4 provide the most energy saving (11.68×) with the least performance degradation (sensitivity of 100 percent, latency of 4.37 seconds, and false alarm rate of 0.15 per hour).

4.1.3 *Error-Aware Inference Model*

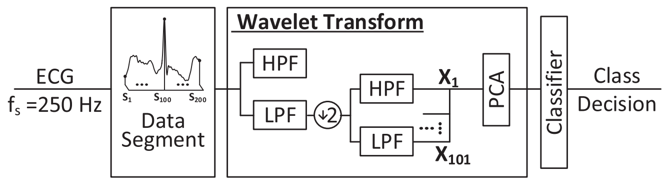
Next, we analyze the effects of error-aware and non-error-aware inference shown in Figs. 8b, 8c, and Fig. 8d. Without error-aware inference, the performance degrades significantly with decreasing feature quality. Using the parameters for the best-performing set of models mentioned earlier (square root and subtraction, genes: 20, depth: 6), the performance without error-aware inference is 100 percent sensitivity, 4.12 seconds latency, and 0.23 false alarms per hour. However, a larger performance difference between error-aware and non-error-aware inference results is observed for GP models with lower feature qualities. As the feature qualities drop below 65.00 percent, the ML systems without error-aware inference models result in false-alarm rates that are an order of magnitude higher than those with error-aware inference models. For example, for the GP model with 1 gene and gene depth of 5, the ML system without error-aware inference model achieves 100 percent sensitivity, 7.36 seconds latency, and 2.12 false alarms per hour. The latency increases by 1.81 seconds compared to the accuracy achieved by error-aware inference. The number of false alarms is an order of magnitude higher than that of the error-aware inference model. Thus, by employing error-aware inference, we are able to recover performance from lower feature qualities, while still offering low FE energy.

In Fig. 9, we demonstrate the impact of the error-aware inference model. Since the number of feature dimensions is quite high (432), we use principal component analysis (PCA) to reduce the number of dimensions to two for visualization purposes. PCA is not used for the FE stage of this application. Fig. 9a shows features from the baseline FE method (which includes down-sampling, filtering, and energy accumulation) as well as the decision boundary. Figs. 9b and c show features from the high-energy GP models (1.33mJ per FV) and low-energy GP models (0.08mJ per FV), respectively. The two sets of GP models offer FE energy reduction between the extremes of 3.60× (for the high-energy GP model) and 59.88× (for the low-energy GP model) with respect to the baseline. As shown, the feature distributions for the two sets of GP models get altered. If the decision boundary from baseline features (without error-aware inference), drawn as a dashed line, is used, it would misclassify the seizure features from both the GP models. On the other hand, error-aware inference, based on features approximated by the GP models (drawn with a solid line), suggests that classification performance can be restored.

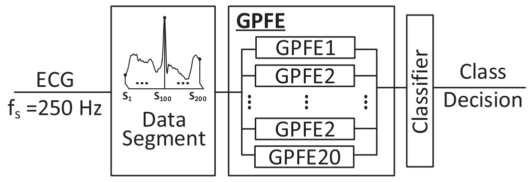
4.2 Case 2: ECG-Based Arrhythmia Detection

In our second case study, we use 38 segments of ambulatory ECG recordings from the MIT-BIH Arrhythmia Database for arrhythmia detection [42], [46]. Each recording contains arrhythmia and normal heart beats, independently annotated by two or more cardiologists. The application is based on the algorithm presented by Martis et al. [47]. 200 samples centered at the R-peak of each beat are used to extract an FV using 4-level discrete wavelet transform (DWT) with Daubechies wavelet (DB-4), as shown in Fig. 10a. The FV consists of 101 features. PCA is then used to reduce the FV dimension to 20 by selecting the feature dimensions with the highest variances. The selected 20 features are passed to a polynomial SVM for classification. Thus, the FE stage consists of two steps: (i) DWT, and (ii) PCA. We model both steps through GP, as shown in Fig. 10b. Thus, for the FE step, 20 GP models are produced.

We evaluate arrhythmia detection system performance through sensitivity (the percentage of arrhythmias correctly identified out of all arrhythmia-labeled data), specificity (the percentage of normal heartbeats correctly identified out of all normal-labeled data), and accuracy (the percentage of correctly identified data).



(a)



(b)

Fig. 10. System diagram for ECG-based arrhythmia detection: (a) Baseline approach using DWT and PCA for FE, and (b) GP-driven FE.

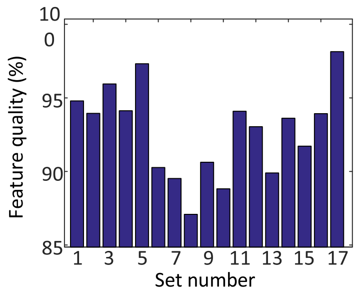
4.2.1 GP Model Selection

For the arrhythmia detection system, we consider single PFs (those given in Table 1) and a select combination of two PFs, with 20 genes and a gene depth of 7. Fig. 11 shows the feature qualities of the different GP models. The type and combination of PFs do not seem to affect the feature quality as greatly as in the case of the EEG application. This similarity suggests a relative ease in fitting GP models for the ECG application dataset compared to the EEG application dataset.

In picking the best models, we consider both the feature quality as well as the energy required for the GP models to approximate the FV. From Fig. 11, we can see that the set of models with the highest feature quality (98.14 percent) uses logarithm and square root PFs. The second best set uses the exponential PF and results in a feature quality of 97.37 percent. The set of models based on multiplication has the third highest feature quality (95.95 percent). Based on feature quality alone, we would choose the first set of models since it has the highest average feature quality. However, the first two sets involve computation-expensive PFs: logarithm and exponential. From MSP430 simulation results, we found that multiplication takes only 10 percent of the cycles compared to exponential or logarithm functions. Taking into account both computational complexity and feature quality, we decided to use the third best set with the multiplication function. In doing so, we can potentially reduce the number of cycles by 10× while loosing less than 3.00 percent feature quality compared to choosing the set with the highest feature quality.

4.2.2 System Analysis

For system analysis, we evaluate the energy required for GP-driven FE to extract the FV from ECG signals. We use multiplication as the PF, with all combinations of number of genes (1, 5, 10, 20) and gene depth (2 to 7). We ran the algorithm five times for all patients and present the average performance with respective standard deviations in Table 5. Since each run contains 38 patients' results, we also conducted two-tailed t-tests to see if the accuracies for GP-driven features with/without error-aware inference are significantly different from baseline performance. The baseline performance is calculated using baseline features (extracted using DWT and PCA). The average performance of these runs is also shown in Fig. 12.



(a)

Set number	1 PF	Set number	2 PFs
1	sqrt	8	sqrt,-
2	log	9	log,-
3	*	10	log,+
4	sqr	11	sqrt,*
5	exp	12	sqrt,sqr
6	-	13	log,exp
7	+	14	sqrt,log
		15	exp,*
		16	log,*
		17	log,sqr

(b)

Fig. 11. Feature quality versus PFs for ECG-based arrhythmia detection: (a) Feature quality of different sets of GP models, and (b) PFs involved in GP model sets 1 to 7 with one PF and sets 8 to 17 with two PFs.

Fig. 12 shows the feature quality and classification results for the selected sets of GP models. The baseline sensitivity, specificity, and accuracy are 82.05, 88.12, and 87.92 percent, respectively, at 11.69μJ per FV.

Fig. 12a shows how the feature quality scales with respect to GP-driven FE energy. Both the feature quality and FE energy show distinct step patterns as opposed to the smooth degradation we saw earlier in Fig. 8a. In this application, the feature qualities of the GP models do not improve much with increasing gene depth. For example, for a given number of genes, the feature quality varies within 4.00 percent from depth 2 to depth 7. The fact that the number of genes has a greater impact on feature quality than gene depth suggests that the features can be well approximated with simple multiplications. In other words, multi-level multiplications do not improve the feature quality of approximated features.

As can be seen from Fig. 12a, the overall FE energy can be halved before feature quality drops by more than 6.00 percent. This suggests the existence of a group of branches and trees that do not contribute much to overall GP model performance (i.e., additional computations do not improve feature quality much). For example, at a gene depth of 7, the FE energy decreases from 12.47μJ per FV (20 genes) to 6.54μJ per FV (10 genes). However, the average feature quality does not degrade drastically (only from 98.87 to 89.62 percent). Therefore, we can pick the models with lower FE energy without incurring much performance degradation (note that we can use error-aware inference to recover from the degradation).

Figs. 12b, 12c, and Fig. 12d present the sensitivity, specificity, and accuracy for the selected GP models. The sensitivity is kept within 1.25 percent of baseline sensitivity (82.05 percent). This enables us to compare the degradation in specificity and accuracy as we scale the GP-driven FE energy. Consider the GP model that requires the highest FE energy (13.43μJ per FV at 20 genes and depth of 3) with specificity and accuracy of 87.30 and 87.16 percent, respectively. These are within 1.00 percent of the baseline performance (88.12 specificity, 87.92 percent accuracy). The accuracy is not significantly different from the baseline accuracy, as can be seen from Table 5. This performance can be maintained (84.85 specificity, 85.08 percent accuracy, not significantly different from baseline accuracy) even as we scale FE energy down to 6.49μJ per FV using 10 genes and a gene depth of 2. If we can tolerate some performance degradation, we can choose the set of GP models derived with 1 gene and a gene depth of 2 (sensitivity: 81.17 percent, specificity: 80.63 percent, accuracy: 81.86 percent). The FE

TABLE 5
GP Model Results for the Arrhythmia Detection Application

					Error-aware inference results						Non-error-aware inference results					
Number of genes	Gene depth	GP-driven FE energy (μJ/FV)	Feature quality (%)		Sensitivity (%)		Specificity (%)		Accuracy (%)		Sensitivity (%)		Specificity (%)		Accuracy (%)	
			Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	**2**	**1.13**	**20.13**	**0.00**	**81.17**	**0.13**	**80.63**	**0.04**	**81.86**	**0.03**	**81.05**	**0.05**	**64.73**	**0.16**	**68.17*****	**0.52**
	3	1.37	20.23	0.09	80.71	0.32	80.71	0.34	81.82	0.30	81.20	0.07	64.55	0.08	68.19***	0.06
	4	1.74	20.19	0.11	81.15	0.52	80.33	0.18	81.45*	0.16	81.24	0.03	64.50	0.05	68.16***	0.03
	5	1.55	20.12	0.09	81.35	0.44	80.21	0.26	81.46*	0.14	81.41	0.44	64.51	0.10	68.17***	0.07
	6	1.60	20.23	0.09	80.91	0.26	80.47	0.30	81.37*	0.16	81.23	0.03	64.52	0.04	68.17***	0.02
	7	1.85	20.23	0.09	81.46	0.25	80.39	0.43	81.58*	0.25	81.23	0.03	64.52	0.04	68.17***	0.02
5	2	3.52	68.72	0.80	80.76	0.71	81.01	0.55	81.69	0.37	81.32	0.46	81.52	1.22	81.56*	0.63
	3	3.91	69.43	1.21	81.41	0.48	81.37	0.81	82.09	0.99	81.50	0.12	79.53	1.10	79.94*	0.95
	4	3.51	68.55	0.83	80.96	0.42	81.80	0.89	82.44*	0.96	81.38	0.84	79.28	1.57	79.81**	1.26
	5	3.67	69.08	0.62	80.70	0.85	81.64	0.58	82.16	0.55	81.14	0.61	81.05	1.49	81.18	1.30
	6	4.11	68.92	0.55	80.90	0.51	81.84	1.06	82.40	1.05	81.18	1.07	80.93	1.68	80.88**	1.03
	7	3.63	68.42	0.92	80.50	0.36	80.57	1.22	81.01*	1.38	80.52	0.50	80.19	1.81	80.32*	1.31
10	2	6.49	89.93	0.76	81.00	0.57	84.85	0.64	85.08	0.52	81.59	0.91	83.20	0.79	83.09	0.74
	3	**6.93**	**89.85**	**1.26**	**81.68**	**0.78**	**84.92**	**1.13**	**85.16**	**0.88**	**81.72**	**0.51**	**84.27**	**1.36**	**84.04**	**1.09**
	4	6.60	89.98	1.49	81.61	0.60	84.27	0.60	84.63	0.52	81.39	0.38	84.23	1.34	83.78	1.02
	5	6.88	90.08	1.01	81.33	0.82	85.48	1.29	85.58	1.11	81.88	0.61	83.29	0.59	83.18	0.29
	6	6.83	90.01	1.25	81.25	0.83	84.75	1.02	85.01	0.93	81.28	0.27	83.52	0.63	83.50	0.58
	7	6.54	89.62	0.83	80.85	0.44	85.14	1.54	85.25	1.29	81.95	0.48	83.98	0.93	83.82	0.71
20	2	12.34	99.04	0.09	81.45	1.09	87.41	0.97	87.15	0.84	82.33	0.48	84.87	0.30	84.78	0.08
	3	13.43	98.94	0.15	82.11	0.62	87.30	1.38	87.16	1.11	82.36	0.41	85.43	0.67	85.19	0.67
	4	12.91	98.98	0.05	81.44	0.49	87.74	0.78	87.54	0.72	81.81	0.95	85.14	0.43	84.82	0.15
	5	12.78	98.97	0.09	82.13	0.79	87.50	0.55	87.24	0.46	82.13	0.90	85.42	0.83	85.10	0.62
	6	12.87	98.78	0.17	81.63	0.42	87.46	0.90	87.21	0.86	82.37	0.80	84.57	0.80	84.37	0.97
	7	12.47	98.87	0.14	81.90	0.28	88.11	0.47	87.92	0.52	82.07	1.14	85.47	0.32	85.17	0.27

GP-driven FE energy is presented with the average results and standard deviations of feature quality, error-aware inference results, and non-error-aware inference results. Bold results correspond to the high-energy and low-energy models mentioned in the text.
** p-value <0.05, ** p-value <0.01, *** p-value <0.001*

energy is now only $1.13\mu J$ per FV. This is a $10.35\times$ reduction relative to the baseline FE energy of $11.69\mu J$ per FV.

4.2.3 Error-Aware Inference Model

We further consider the effects of error-aware inference by comparing the error-aware inference results with the non-error-aware inference results. When the error-aware inference model is not used, the specificity and accuracy degrade with feature quality. For GP models with feature qualities over 80.00 percent, both specificity and accuracy are within 6.00 percent of the baseline performance even without the use of error-aware inference. However, for models with feature qualities below 80.00 percent, the specificity and accuracy of the lowest energy model ($1.13\mu J$ per FV) degrade to 64.73 and 68.17 percent (significantly different from baseline accuracy with $p < 0.001$), respectively.

We show the impact of error-aware inference models in Fig. 13. Each plot depicts actual features. Since the number of feature dimensions (20) is high, PCA is again used to aid visualization. Fig. 13a shows features from the baseline FE method (DWT and PCA) as well as the decision boundary. Figs. 13b and 13c show the features from the high-energy GP models ($6.93\mu J$ per FV at 10 genes and depth of 3) and low-energy GP models ($1.13\mu J$ per FV at 1 gene and depth of 2), respectively. As can be seen, the feature distribution of the low-energy GP-approximated features is altered more than that of the high-energy GP-approximated features. The decision boundary from baseline features (without error-aware inference), drawn as a dashed line, misclassifies the arrhythmia features for the low-energy GP model. However, the error-aware inference model, constructed from the low-energy GP-approximated features (drawn as a solid line), suggests that classification performance can be restored. The accuracy without error-aware inference is 68.17 percent (significantly different from baseline accuracy with $p < 0.001$) and can be restored to 81.86 percent (not that different from baseline accuracy) with error-aware inference.

The high-energy GP-approximated features [Fig. 13b] have feature distributions that are close to the baseline feature distributions. Thus, we see similar accuracies (difference of 1.12 percent) with or without the error-aware inference models (both accuracies are not significantly different from baseline accuracy).

4.3 Classification Energy

Table 6 shows the baseline FE energy, GP-driven FE energy, and classification energy (with accelerator) for each case study.

For the seizure detection application, RBF-SVM is used as the classifier. The classification energy of each test FV is dependent on the number of support vectors in the case of RBF-SVM. We need $45.10\mu J$ per FV classification for an inference model built with baseline FVs. The classification energy ranges from $42.11\mu J$ to $51.34\mu J$ per FV classification for error-aware inference models built with GP-approximated FVs. For this application, the SV Mem. is not large

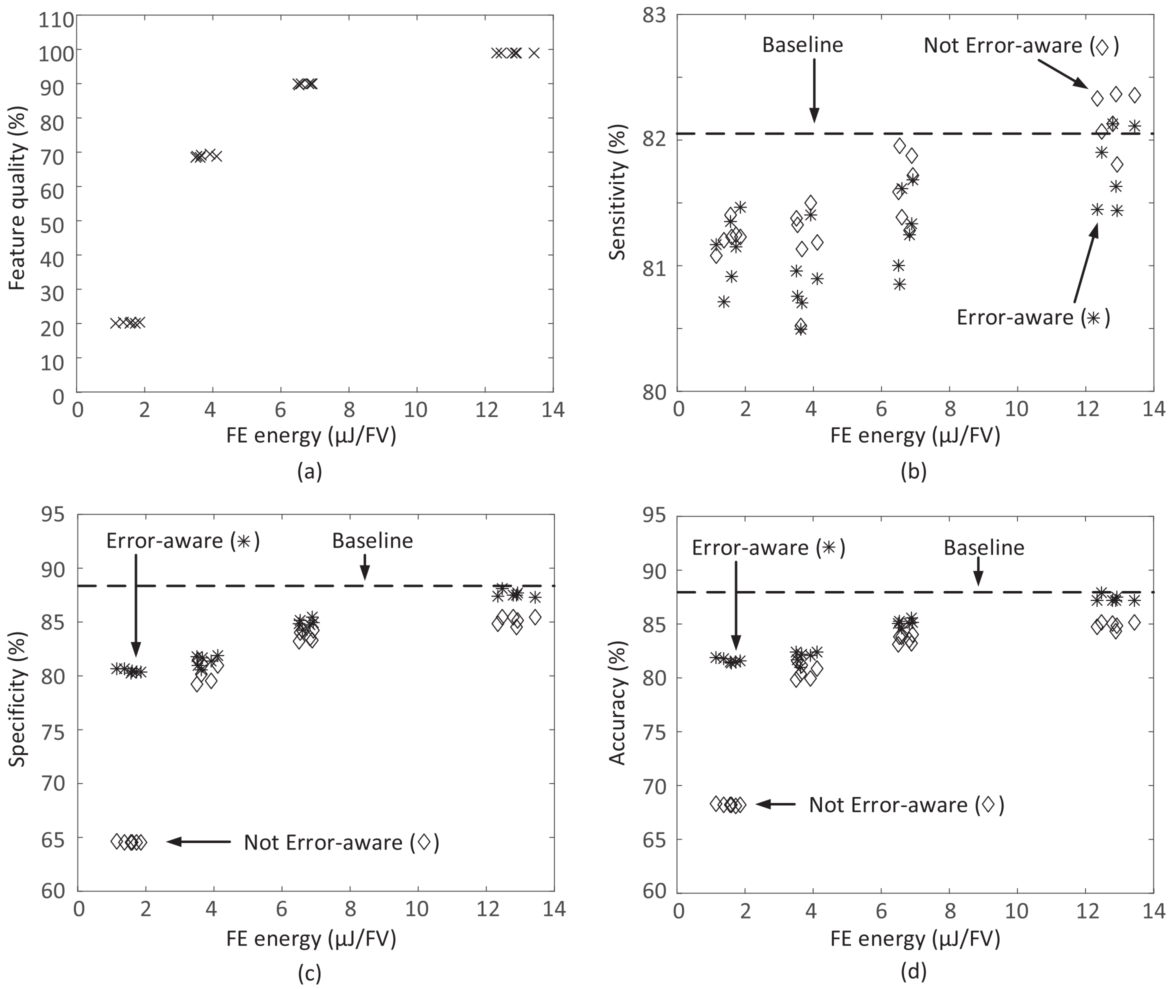


Fig. 12. Arrhythmia detection performance with respect to FE energy: (a) Feature quality versus FE energy, (b) sensitivity versus FE energy, (c) specificity versus FE energy, (d) accuracy versus FE energy.

enough to hold all the support vectors. Since the feature dimension is 432 and the inference model can use up to 250 support vectors for this application, at least 216 KB memory is needed. To alleviate this memory requirement, we can use a compression-decompression accelerator (CDA), which has been provided on this microprocessor [38]. The CDA consists of two parallel encoders and decoders and supports 16b data formats. It can realize compressions up to 4×. Thus, with the help of a CDA, a 64 KB memory should suffice to fully implement RBF-SVM for the EEG seizure detection application.

For the arrhythmia detection application, we use a 2nd-order polynomial SVM as the classifier. We transform the SVM support vectors into matrix form in order to realize a linear computation of the quadratic polynomials [14]. This allows us to negate any dependence of classification energy on the number of support vectors within the inference model. This method is only beneficial for polynomial SVMs with a large number of support vectors and a small feature dimension (i.e., the number of features). This is because the matrix involved has a dimension of $(M+1)\times(M+1)$, where M is the feature dimension. Therefore, as the feature dimension increases, the number of computations increases quadratically. For our application, the feature dimension is 20, and the number of support vectors is over 1000. Therefore, such a formulation can significantly reduce the amount of computations required to implement the polynomial SVM. In this way, the classification energy can be reduced to just $0.24\mu J$ per FV classification for both error-aware and non-error-aware inference models.

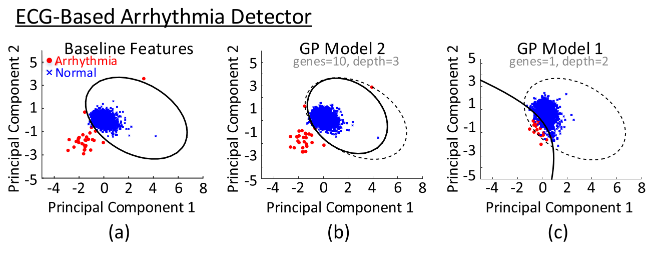


Fig. 13. Polynomial SVM classifier boundary for (a) baseline features, (b) GP-approximated features (PF: multiplication, genes: 10, depth: 3), and (c) GP-approximated features (PF: multiplication, gene: 1, depth: 2).

TABLE 6
Comparison of Baseline FE Energy and GP-Driven FE Energy for EEG-Based Seizure Detection and ECG-Based Arrhythmia Detection

	Seizure Detection	Arrhythmia Detection
FE	4.79 mJ	11.69 μJ
GP-driven FE	0.02-1.59 mJ	1.13-13.43 μJ
Classification	45.10 μJ	0.24 μJ

Classification energy per FV for baseline FE is also shown for each application.

In both applications, the FE energy is the dominant part of overall system energy. Therefore, by reducing the FE energy while maintaining a similar classification energy, we can reduce the overall system energy by 10.62× for seizure detection and 8.70× for arrhythmia detection.

5 Conclusions

In this paper, we presented an approach that makes synergistic use of GP and error-aware inference to implement ML applications very energy-efficiently while maintaining system performance. We evaluated the approach with two applications. For EEG-based seizure detection, our approach achieves perfect sensitivity, 4.37 seconds latency, and 0.15 false alarms per hour at 11.68× reduction in FE energy. This compares with a baseline performance of perfect sensitivity, 3.84 seconds latency, and 0.06 false alarms per hour. For ECG-based arrhythmia detection, we achieve 81.17 percent sensitivity, 80.63 percent specificity, and 81.86 percent accuracy at 10.35× FE energy saving. This compares with a baseline performance of 82.05 percent sensitivity, 88.12 percent specificity, and 87.92 percent accuracy. The proposed approach thus enables easy and flexible implementation of ML systems at very low energy and acceptable system performance. In future work, we we will evaluate Cartesian GP, which enables evolution of multiple features to be completed in parallel [48]. In addition, we will consider improvements in fitness functions to achieve quicker fitness calculation and avoid the need for baseline features. One potential area for exploration is fitness functions aiming to maximize feature generalization as well as the empirical mutual information between features and output classes.

Acknowledgments

This work was supported by NSF (CNS-1617640) as well as AFOSR (FA9550-14-1-0293).

References

[1] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction," in *Proc. 36th Int. Symp. Microarchitecture*, Dec. 2003, pp. 81–93.

[2] V. Saripalli, G. Sun, A. Mishra, Y. Xie, S. Datta, and V. Narayana, "Exploiting heterogeneity for energy efficiency in chip multiprocessors," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 2, pp. 109–119, Jul. 2011.

[3] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 449–460.

[4] V. Chippa, D. Mohapatra, K. Roy, S. Chakradhar, and A. Raghunathan, "Scalable effort hardware design," *IEEE Trans. VLSI Syst.*, vol. 22, no. 9, pp. 2004–2016, Sep. 2014.

[5] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *Proc. Design, Automat. Test Europe Conf.*, Mar. 2015, pp. 701–796.

[6] C. Alvarez, J. Corbal, and M. Valero, "Fuzzy memoization for floating-point multimedia applications," *IEEE Trans. Comput.*, vol. 54, no. 7, pp. 922–927, Jul. 2005.

[7] J. R. Koza, "Detailed description of genetic programming," in *Genetic Programming: On the Programing of Computers by Means of Natural Selection*. Cambridge, MA, USA: A Bradford Book, MIT Press, 1992, pp. 17–62.

[8] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2012, pp. 5285–5288.

[9] Z. Wang, R. Schapire, and N. Verma, "Error adaptive classifier boosting (EACB): Leveraging data-driven training towards hardware resilience for signal inference," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 4, pp. 1136–1145, Apr. 2015.

[10] H. Guo, L. Jack, and A. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Trans. Syst. Man Cybern.*, vol. 35, no. 1, pp. 89–99, Feb. 2005.

[11] J. Kishore, L. Patnaik, V. Mani, and V. Agrawal, "Application of genetic programming for multicategory pattern classification," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 242–258, Sep. 2000.

[12] A. H. Gandomi and A. H. Alavi, "A new multi-gene genetic programming approach to nonlinear system modeling. part I: Materials and structural engineering problems," *Neural Comput. Appl.*, vol. 21, no. 1, pp. 171–187, Feb. 2012.

[13] H. Jia, J. Lu, N. K. Jha, and N. Verma, "A heterogeneous microprocessor for energy-scalable sensor inference using genetic programming," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2017, pp. C28–C29.

[14] 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," *IEEE J. Solid-State Circuits*, vol. 48, no. 7, pp. 1625–1637, Jul. 2013.

[15] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surveys*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016.

[16] L. Perterson and B. Davie, *Computer Networks, 5th Edition: A Systems Approach*. Burlington, MA, USA: Morgan Kaufmann Publishers, 2011.

[17] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. Des. Automat. Conf.*, Jun. 2012, pp. 820–825.

[18] J. Miguel, M. Badr, and N. Jerger, "Load value approximation," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2013, pp. 127–139.

[19] J. Park, J. Choi, and K. Roy, "Dynamic bit-width adaptation in DCT: An approach to trade off image quality and computation energy," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 5, pp. 787–793, Jun. 2010.

[20] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *Proc. Design Automat. Conf.*, Jun. 2010, pp. 555–560.

[21] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recog.*, vol. 35, no. 6, pp. 1197–1208, Jun. 2002.

[22] P. A. Estevez, N. Becerra-Yoma, N. Boric, and J. A. Ramirez, "Genetic programming-based voice activity detection," *Electron. Lett.*, vol. 41, no. 20, pp. 1141–1143, Sep. 2005.

[23] Z. Wang, K. H. Lee, and N. Verma, "Overcoming computational errors in sensing platforms through embedded machine-learning kernels," *IEEE Trans. VLSI Syst.*, vol. 23, no. 8, pp. 1459–1470, Aug. 2015.

[24] T. von Sydow, B. Neumann, H. Blume, and T. G. Noll, "Quantitative analysis of embedded FPGA-architectures for arithmetic," in *Proc. 17th Int. Conf. Appl.-Specific Syst., Archit. Processors*, Sep. 2006, pp. 125–131.

[25] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Highly energy-efficient and quality-tunable inexact FFT accelerators," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2014, pp. 1–4.

[26] T. Chen, et al., "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 269–284, Feb. 2014.

[27] J. Struharik, "Implementing decision trees in hardware," in *Proc. IEEE Int. Symp. Intell. Syst. Inform.*, Sep. 2011, pp. 41–46.

[28] Z. Xie, T. Quirino, M. L. Shyu, and S. C. Chen, "ASIC: Supervised multi-class classification using adaptive selection of information components," in *Proc. Int. Conf. Semantic Comput.*, Sep. 2007, pp. 527–534.

[29] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum, "The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming," in *Proc. 2nd Int. Conf. Genetic Programming*, Jul. 1997, pp. 304–312.

[30] M. Kotani, S. Ozawa, M. Nakai, and K. Akazawa, "Emergence of feature extraction function using genetic programming," in *Proc. 3rd Int. Conf. Knowl.-Based Intell. Inf. Eng. Syst.*, Dec. 1999, pp. 149–152.

[31] D. Y. Harvey and M. D. Todd, "Automated feature design for numeric sequence classification by genetic programming," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 474–489, Aug. 2015.

[32] U. Bhowan, M. Johnston, M. J. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 368–386, Jun. 2013.

[33] K. Nag and N. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 499–510, Feb. 2016.

[34] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evol. Comput.*, vol. 5, no. 4, pp. 373–399, 1997.

[35] K. O. Stanley and R. Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2002, pp. 569–577.

[36] N. Garcia-Pedrajas, C. Hervás-Martinez, and J. Muñoz-Pérez, "COVNET: A cooperative coevolutionary model for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 575–596, May 2003.

[37] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications*. Berlin, Germany: Springer, 2008, vol. 207.

[38] K. H. Lee and N. Verma, "A low-power microprocessor for data-driven analysis of analytically-intractable physiological signals in advanced medical sensors," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2013, pp. C250–C251.

[39] IAR Systems, "MSP430 IAR Embedded Workbench® IDE project management and building guide," IAR Systems AB, Uppsala, Uppsala County, Sweden, Tech. Rep. SLAA329, Nov. 2015.

[40] K. Venkat, "Efficient multiplication and division using MSP30," Texas Instruments Incorporated, Dallas, Texas, USA, Tech. Rep., Sep. 2006.

[41] Texas Instruments, "IAR Embedded WorkbenchTM version 3+ for MSP430TM," Texas Instruments Incorporated, Dallas, Texas, USA, Tech. Rep., Jun. 2016.

[42] A. L. Goldberger, et al.,"Physiobank, Physiotoolkit, and Physionet components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, Jun. 2000.

[43] M. Srinivas and M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Sys., Man Cyb.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.

[44] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, "Troubleshooting GP, " in *A Field Guide to Genetic Programming*. Raleigh, NC, USA: Lulu. com, 2008.

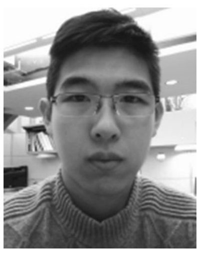
[45] A. Parkins and A. K. Nandi, "Genetic programming techniques for hand written digit recognition," *Signal Process.*, vol. 84, no. 12, pp. 2345–2365, Jul. 2004.

[46] G. Moody and R. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Eng. Med. Biol.*, vol. 20, no. 3, pp. 45–50, Jun. 2001.

[47] R. Martis, et al., "Automated screening of arrhythmia using wavelet based machine learning techniques," *J. Med. Syst.*, vol. 36, no. 2, pp. 677–688, Apr. 2012.

[48] J. Miller and P. Thomson, "Cartesian genetic programming," in *Proc. Eur. Conf. Genetic Programming*, Apr. 2000, pp. 121–132.

Jie Lu received the BSc (Hons.) degree in life sciences from Queens University, Kingston, Canada, in 2010, and the MASc degree in biomedical engineering from the University of Toronto, Toronto, Canada, in 2013, and the MA degree in electrical engineering from Princeton University, Princeton, in 2015. Currently, she is working toward the PhD degree in electrical engineering with Princeton University. Her main area of research interests include energy-efficient platforms for sensing and computing. She has presented her work at the 2013 International BCI Meeting and published in Frontiers on Brain-computer Interfaces and *IEEE Transactions on Computers*. During her masters and undergraduate degrees, she received James F. Crothers Fellowships in Peripheral Nerve damage 2012, Kimel Family Graduate Student Scholarship 2012, and NSERC-USRA 2008. She is a student member of the IEEE.



Hongyang Jia received the BEng degree in microelectronics from Tsinghua University, Beijing, China, in 2014, and the MA degree in electrical engineering from Princeton University, Princeton, in 2016, where he is currently working toward the PhD degree. His research focuses on ultra-low energy system design for inference application. His primary research interests include CMOS IC design, which leverages approximate computing techniques for model complexity reduction, automatic program synthesis, and mapping for feature extraction on various sensing platforms. He received Analog Devices Outstanding Student Designer Award in 2017. He is a student member of the IEEE.

Naveen Verma received the BASc degree in electrical and computer engineering from the University of British Columbia, Vancouver, Canada in 2003, the MS and the PhD degrees in electrical engineering from the Massachusetts Institute of Technology, in 2005 and 2009 respectively. Since July 2009, he has been in the Department of Electrical Engineering, Princeton University, where he is currently an associate professor. His research focuses on advanced sensing systems, including low-voltage digital logic and SRAMs, low-noise analog instrumentation and data-conversion, large-area sensing systems based on flexible electronics, and low-energy algorithms for embedded inference, especially for medical applications. He is a Distinguished Lecturer of the IEEE Solid-State Circuits Society, and serves on the technical program committees for ISSCC, VLSI Symp., DATE, and IEEE Signal-Processing Society (DISPS). He is the recipient or co-recipient of the 2006 DAC/ISSCC Student Design Contest Award, 2008 ISSCC Jack Kilby Paper Award, 2012 Alfred Rheinstein Junior Faculty Award, 2013 NSF CAREER Award, 2013 Intel Early Career Award, 2013 Walter C. Johnson Prize for Teaching Excellence, 2013 VLSI Symp. Best Student Paper Award, 2014 AFOSR Young Investigator Award, 2015 Princeton Engineering Council Excellence in Teaching Award, and 2015 IEEE Trans. CPMT Best Paper Award. He is a member of the IEEE.

Niraj K. Jha (S'85-M'85-SM'93-F'98) received the BTech degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India in 1981, the MS degree in electrical engineering from the S.U. N.Y., Stony Brook, NY, in 1982, and the PhD degree in electrical engineering from the University of Illinois, Urbana-Champaign, IL, in 1985. He is a professor of electrical engineering with Princeton University. He has served as the editor-in-chief of the *IEEE Transactions on VLSI Systems* and an associate editor of the *IEEE Transactions on Circuits and Systems I and II*, the *IEEE Transactions on VLSI Systems*, the *IEEE Transactions on Computer-Aided Design*, the *IEEE Transactions on Computers*, the *Journal of Electronic Testing: Theory and Applications*, and the *Journal of Nanotechnology*. He is currently serving as an associate editor of the *IEEE Transactions on Multi-Scale Computing Systems* and the *Journal of Low-Power Electronics*. He has served as the Program Chairman of the 1992 Workshop on Fault-Tolerant Parallel and Distributed Systems, the 2004 International Conference on Embedded and Ubiquitous Computing, and the 2010 International Conference on VLSI Design. He has served as the director of the Center for Embedded System-on-a-chip Design funded by New Jersey Commission on Science and Technology and the associate director of the Andlinger Center for Energy and the Environment. He is the recipient of the AT&T Foundation Award and NEC Preceptorship Award for research excellence, NCR Award for teaching excellence, Princeton University Graduate Mentoring Award, and six Outstanding Teaching Commendations. He received the Distinguished Alumnus Award from I.I.T., Kharagpur in 2014. He has co-authored or co-edited five books that include two textbooks: *Testing of Digital Systems* (Cambridge University Press, 2003) and *Switching and Finite Automata Theory, 3rd edition* (Cambridge University Press, 2009). He has also co-authored 15 book chapters and more than 430 technical papers. He has coauthored 14 papers that have won various awards, and another six that have received best paper award nominations. He has received 17 U.S. patents. He has served on the program committees of more than 170 conferences and workshops. His research interests include embedded computing, secure computing, machine learning, IoT, low power hardware/software design, and computer-aided design of integrated circuits and systems. He has given several keynote speeches in the areas of nanoelectronic design/test, embedded system security, and smart healthcare. He is a fellow of the IEEE and the ACM.