A Heterogeneous Microprocessor for Energy-scalable Sensor Inference Using Genetic Programming

Hongyang Jia, Jie Lu, Niraj K. Jha, and Naveen Verma
Princeton University, Princeton, NJ, USA (hjia@princeton.edu)

Abstract

We present a heterogeneous microprocessor for IoE sensor-inference applications, which achieves programmability required for feature extraction strictly using application data. Acceleration, though key for energy efficiency, poses substantial programmability challenges. These are overcome by exploiting genetic programming (GP) for automatic program synthesis. GP yields highly structured models of computation, enabling: (1) high degree of specialization; (2) systematic mapping of programs to the accelerator; and (3) energy scalability via user-controllable approximation. The microprocessor (130nm) achieves 325×/156× energy reduction, and further 20×/9× energy scalability, for programmable feature extraction in two medical-sensor applications (seizure/arrhythmia-detection) vs. GP-model execution on CPU. The energy efficiency is 220 GOPS/W, near that of fixed-function accelerators, exceeding typical programmable accelerators.

System Overview

Fig. 1a (top) shows a generic sensor-inference system, consisting of feature-extraction and classification stages. Classification requires specific computational kernels from machine learning [support-vector machines (SVM), neural networks, decision trees], and is readily delegated to energy-efficient accelerators. This typically leaves feature-extraction energy to dominate, as shown in Fig. 1b for an ECG-based arrhythmia detector [1]. The challenge is that feature extraction varies greatly across applications and signals. Though acceleration is one of the most significant ways to enhance energy efficiency, its major problem is programmability, on several levels: (1) reducing specialization degrades energy efficiency; (2) mapping programs to harness specialization is extremely difficult without intimate knowledge of the accelerator microarchitecture; (3) only weak knobs are typically available for energy scalability, especially exploiting approximate computing, which shows significant promise in inference applications.

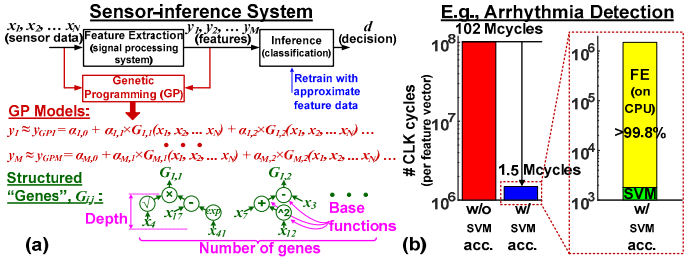


Fig. 1: (a) Overview of genetic programming for sensor-inference applications, and (b) energy breakdown in app. using classification accelerator.

While GP has previously been employed for automatic program synthesis from high-level specifications (i.e., input-output data), here it is exploited to enable a heterogeneous architecture overcoming these challenges. Fig. 1a shows the key attributes of GP made use of. GP takes inputs $x_1 \dots x_N$ and outputs $y_1 \dots y_M$ (features in the inference system), to construct models of computation. These are referred to as GP models, by which approximations $y_{GP,1} \dots y_{GP,M}$ of the outputs can be derived from the inputs. Importantly, GP models have the specific structure shown, where units called genes $G_{i,j}$, corresponding to a tree of computation applied to inputs, are linearly combined. This structure has enabled learning rules and model generalization so that GP can apply to a broad range of computations. Here, this structure is exploited towards both a high degree of accelerator specialization and systematic mapping of computations to the accelerator. Further, GP proceeds by optimizing a fitness function between y_i and $y_{GP,i}$, under user-provided constraints to gene depth, number, and base functions (Fig. 1a), which all strongly impact energy. Thus, the fitness function provides a formal metric by which users can control an approximation-vs.-energy tradeoff. Here, this tradeoff is enhanced via classifier training, as described later.

Heterogeneous Microprocessor Design

Fig. 2 shows the microprocessor architecture and data flow for sensor inference. A CPU (OpenMSP) is integrated with a programmable Feature-Extraction Accelerator (FEA) [consisting of 4 Gene-Computation (GC) Cores] and an SVM Accelerator (SVMA). Under DMA control, input data is loaded to a Sensor Memory, and accessed by the FEA as requested by GC cores. GC-core outputs feed to a GP Model Manager (GPMM) in the FEA for linear combination, giving features then loaded in an SVM Feature-Vector Buffer. The SVMA performs classification, using support vectors loaded in local memory, and is configurable to various kernels (RBF, poly, linear), vector dimensionalities, and support-vector-set sizes. CPU intervention occurs after interrupt assertion following classification. A Power Management Unit controls fine-grained clock gating, as data moves through the blocks.

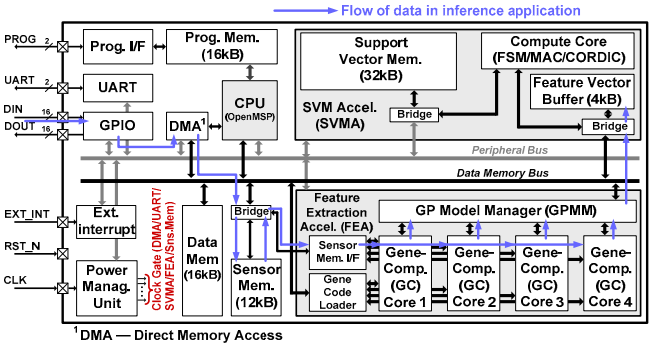
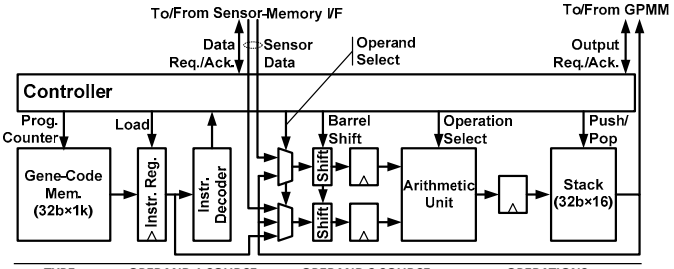


Fig. 2: Architectural diagram of heterogeneous microprocessor.

Fig. 3 shows the GC Core microarchitecture, consisting of Controller, Gene-Code Memory, single-instruction execution pipeline with arithmetic unit [for ADD, SUB, MULT, SQUARE, CORDIC (exp, ln, square-root, reciprocal) operations], and stack scratchpad. The pipeline is optimized to implement tree-structured genes. The 32b gene-code instructions, having types shown, roughly map to a gene-tree node (or a control operation). Supported gene base functions are set by the arithmetic unit, with dynamic-range control enabled by preceding, instruction-configurable barrel shifters.

Classifier Training for Approximation

To enable approximation for greater energy efficiency, retraining of the classification model is performed. In [2], an approach is presented where errors due to hardware faults in the feature-extraction stage are overcome by using the error-affected feature data for classifier training. The resulting model is referred to as an error-aware model. In the proposed



TYPE	OPERAND 1 SOURCE	OPERAND 2 SOURCE	OPERATIONS
NOP	--	--	--
Two-op Instructions	Sensor Memory, Stack	Sensor Memory, Stack, Constant	ADD, SUB, MULT
One-op Instructions	Sensor Memory, Stack	--	Square, CORDIC (expx,lnx,√x,1/x)
Stack	--	Constant	Push, Pop
Control	--	--	Shift, End of gene, et al.

Fig. 3: Microarchitecture of Gene-Computation (GC) Core.

system, energy scalability is achieved by constraining gene depth, number, and base functions. Since this impacts fitness of computed features, an error-aware model is used to restore classification accuracy for graceful degradation. Fig. 4 illustrates error-aware modeling for GP-model approximation, using feature data from the prototype. For two demonstrated applications (described below), the first plot shows data from hand-tuned code (baseline) on the CPU, representing exact features (for visualization, high-dimensional feature data is projected to two dimensions using PCA). The next two plots show data from GP models on the FEA, representing two approximation points. As shown, the feature distributions are significantly altered. But, as is typical, class separability is maintained much more robustly, and the error-aware model substantially restores classification accuracy. Fig. 5 shows the automatic-programming & classifier-training flow developed, taking only input-output feature data and training labels as inputs, providing gene code and classifier model as outputs.

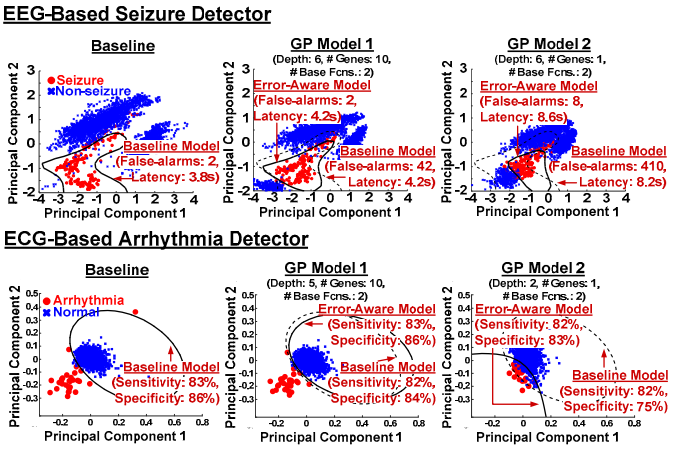


Fig. 4: Error-aware modeling for GP-model approximation.

Prototype Measurements and System Demonstrations

Fig. 6a shows the prototype microprocessor in 130nm CMOS. Two medical-sensor apps are demonstrated: (1) EEG-based seizure detection [3]; and (2) ECG-based cardiac-arrhythmia detection [4]. For variously constrained GP models running on the FEA, Fig. 6b shows the measured feature fitness and classification accuracy (with/without error-aware model) vs. energy. While fitness degrades with energy, error-aware modeling greatly improves classification, enabling a large energy-scalability range (20× and 9.3×).

Fig. 7 shows the measurement summary. Bar plots show feature-extraction energies for the two apps for three cases: (1) baseline implementation on CPU; (2) two GP models (corresponding to Fig. 4) running on CPU; and (3) same GP models running on FEA. On the CPU, even at the high-accuracy approximation points, GP models incur roughly the same energy as baseline implementation (3.5× reduction, 1.1× increase for two apps). But, FEA reduces GP model energies by 325×/293× and 156×/105× for the two apps and approximation points. The summary table shows the block breakdown for energy/clock and energy of the apps, with programmable feature extraction now below fixed-function classification (CPU, DMA are <40nJ/feature-vector). The comparison table shows the FEA energy scalability and efficiency (220 GOPS/W), which is close to fixed-function accelerators, exceeding programmable accelerators, especially when taking technology into consideration. While typical programmable accelerators derive energy-efficiency from vector or array processing, the FEA is enabled by high degree of specialization thanks to GP models.

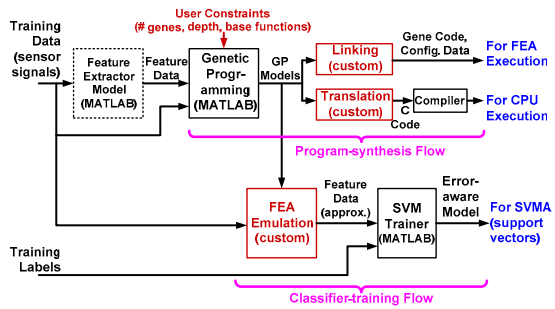


Fig. 5: Automatic-programming and classifier-training flow.

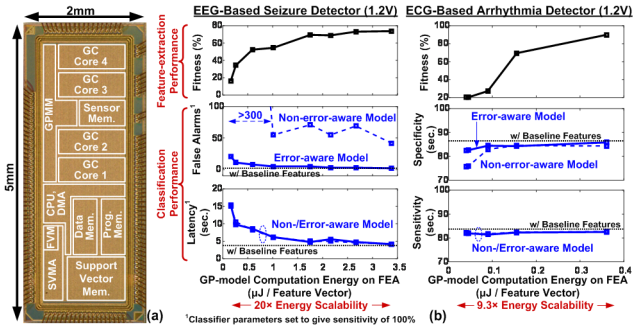
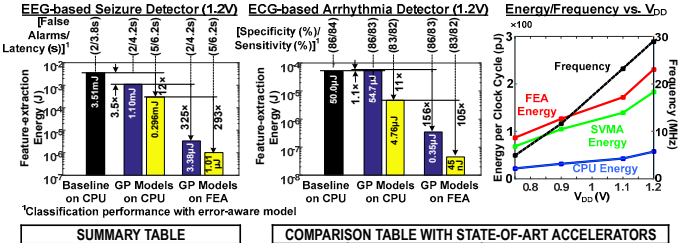


Fig. 6: (a) Die photo of prototype IC implemented in 130nm CMOS, and (b) measured energy-vs.-approximation scalability.



SUMMARY TABLE	
Technology	GF 130nm CMOS
Supply Voltage	0.75 – 1.2V
Clock Frequency	5.0 – 29MHz
Energy/Clock (0.75V):	
(1) CPU	21.8pJ
(2) FEA	86.7pJ
(3) SVMA	68.3pJ
Energy per F.V. (1.2V)²:	FEA/SVMA
(1) Seizure Detect–GP Model 1	3.38µJ / 18.3µJ
(2) Seizure Detect–GP Model 2	1.01µJ / 31.6µJ
(3) Arrhyth. Detect– GP Model 1	0.35µJ / 0.18µJ
(4) Arrhyth. Detect– GP Model 2	0.045µJ / 0.18µJ

²Energy of all other blocks < 40 n.J/F.V.

COMPARISON TABLE WITH STATE-OF-ART ACCELERATORS	FIXED FUNCTION		PROGRAMMABLE		
	Moons VLSI Symp. 2016	Kim ISSCC 2009	GPGPU Nvidia P100	Bohnenstiehl VLSI Symp. 2016	This Work
Technology	40nm	130nm	16nm	32nm	130nm
Quantization Strategy	1 - 16 bit fixed	16 bit fixed	Mixed	16 bit fixed	32 bit fixed
Function	CNN	Obj. Recog.	Prog.	Prog.	Prog.
Energy Scale Strategy	Bit precision	–	–	–	GP model approx.
Energy Scale Range	8.7x	–	–	–	9 – 20x
Energy Efficiency	0.3 - 2.6 TOPS/W	290 GOPS/W	~ 10 GFLOPS/W	172 GOPS/W	221 GOPS/W³

³1 OP equals 1 ADD/MULT

Fig. 7: Measurement summary.

Acknowledgements Funded by AFOSR, NSF (CCF-1253670), C-FAR/SONIC, two SRC STARnet centers by MARCO & DARPA.

References

[1] K. H. Lee, et al., *IEEE JSSC*, July 2013, pp. 1625-1637.
[2] Z. Wang, et al., *IEEE TVLSI*, Aug. 2015, pp. 1459-1470.
[3] A. Shoeb, J. Guttag, *ICML*, June 2010.
[4] E. D. Ubeyli, *Digit. Signal Processing*, May 2007.