Energy-Efficient Classification: Adaptive Approach

Zafar Takhirov, Joseph Wang, Venkatesh Saligrama, Ajay Joshi
Department of Electrical and Computer Engineering, Boston University

***Abstract*—Designing energy-efficient machine learning-based systems for mobile and embedded applications is a hard problem – it is not straight forward to meet the energy-delay constraints, while achieving the accuracy goals. Moreover, the energy, delay and accuracy requirements could change at run-time, which could make the design-time optimization sub-optimal. In this work we propose an adaptive classifier that leverages the wide variability in data complexity to dynamically allocate computational resources and improve energy-efficiency of a hardware accelerator for machine learning. On average, our proposed adaptive classifier can achieve up to $\approx$100$\times$ better energy-efficiency by trading off $\approx$1% accuracy as compared to a complex radial basis function classifier. As compared to a cheap linear classifier our adaptive classifier achieves $\approx$40% higher accuracy, but is $\approx$10$\times$ less energy efficient.**

I. INTRODUCTION

The beginning of the 21st century is proving to be the "data age" – with an exponential increase in mobile and embedded systems, and development of internet, media, and social networks, the amount of data transmitted and processed is booming. At the same time the workloads have become more diverse, and are increasingly trending towards utilizing computationally expensive machine learning approaches to process the large amounts of data [1]. However, small form factors and energy-source limitations of mobile systems, makes execution of machine learning algorithms very challenging. As a result, maintaining a suitable energy dissipation versus performance trade off, while achieving desired accuracy becomes more critical.

There has been some prior work in the design of energy-efficient classifiers. The system proposed by Venkataramani et al. [2] centers on using increasingly complex classifiers on examples close to the decision boundary. This system assumes that complex classifiers will always correctly classify examples, potentially using multiple complex classification functions on inherently noisy examples. Panda et al. [3] introduce a system for object detection using energy-efficient neural computing. Their hierarchical framework of classifiers was set up in an increasing level of complexity, making the dynamic trade-off between classification performance and energy efficiency hard. Similarly, the system proposed by Park et al. [4] uses dynamic threshold adjustment in deep neural networks (DNN) to achieve low power operation. In their work, a bigger DNN is used if smaller DNN fails. Because of the system topology it is not possible to automatically enable the necessary DNN without running all other DNNs.

Our proposed classifier, as described in section II, solves the problem of minimizing the test-time cost [5]. As the main obstacle in the test-time budget problem is the sequential revelation of input information, the optimal energy expenditure of the selection system is generally hard to account for. Additionally, many approaches to test-time budgeted learning [6] require design of both the resource allocation as well as the classification functions, preventing the use of optimized modular systems.

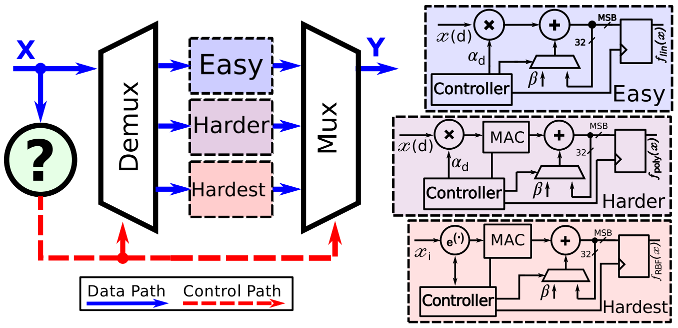

Fig. 1: Conceptual block diagram of the proposed approach. "Chooser" function is marked as ?, and its microarchitecture is the same as the "Easy" block. Data paths and Control paths are shown using solid lines and dashed lines, respectively.

II. ADAPTIVE CLASSIFICATION

Rather than selecting a single classification approach for all examples, our approach is to design an adaptive classification system that dynamically selects a classifier that is appropriate for the data under consideration. In our approach, we leverage the variation in "hardness" of the data across various examples found in real world data sets. In simpler terms, some data samples are extremely hard to classify and require a very complex and energy inefficient classifier to achieve high accuracy. Other input examples could be classified easily and even the simplest classifier would achieve the desired statistical performance at very low energy cost. Intuitively, the strategy of our adaptive classifier system is to route each example to the most energy-efficient classifier such that it is correctly classified, with examples that are incorrectly classified by all decision functions routed to the most energy-efficient classifier. The routing decision is made by a "chooser" function (described in section II), which identifies the hardness of the current input, and routes the input accordingly. Formally, the "hardness" level of any given example is found by solving an unconstrained optimization problem

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} L_j\left(x_i, y_i\right) \mathbb{1}_{g(x_i)=j}, \tag{1}$$

where $L_j\left(x_i, y_i\right) = \mathbb{1}_{f_j(x_i) \neq y_i} + \lambda c_j$ is defined as the loss associated with using the classification function f_j on example x_i. The loss function includes both the classification error $\mathbb{1}_{f_j(x_i) \neq y_i}$ as well as the energy dissipation c_j (λ is a Lagrangian multiplier, $\mathbb{1}_z$ is an indicator function).

	Adaptive Effort Classifier System						Conventional Classifiers					
	High Error Rate		Low Error Rate		Average		Linear		Polynomial		Radial-Basis	
Data Set	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay
Synthetic	52.6e-15	0.1	29e-12	0.7	903.9e-15	0.2	15.8e-15	0.03	92.3e-15	0.3	27e-12	0.4
Image Segm.	4.9e-12	1.0	2.7e-9	4.9	9.3e-12	1.4	5.4e-12	1	79.8e-12	2.6	2.6e-9	4.173
ISOLET	4.5e-9	34.6	2.9e-6	153.4	130.0e-9	69.1	5.9e-9	35.1	79.4e-9	71.4	2.7e-6	142.6
Letter Rec.	4.1e-12	1.0	2.3e-9	4.3	29.7e-12	1.1	4.4e-12	0.8	6.5e-12	2.5	1.9e-9	3.7
MNIST	8.6e-9	43.3	3.9e-6	184.1	186.4e-9	94.8	10.5e-9	44.5	14e-9	86.8	3.93e-6	181.3
Penbase Rec.	3.3e-12	0.7	2.0e-9	3.7	21.5e-12	1.5	4e-12	0.9	5.3e-12	1.6	1.8e-9	3.7
Spam filter	50.1e-12	2.9	21.6e-9	14.5	1.1e-9	4.3	53.1e-12	3.2	67.6e-12	5.4	20.5e-9	13.02
Vowel Rec.	1.4e-12	0.5	73.2e-11	2.8	15.9e-12	1.2	1.7e-12	0.5	2.4e-12	1.3	0.7e-9	2.3

TABLE I: Energy and delay for the adaptive and conventional classifiers. Energy results are shown in J/cycle and delay is represented in μs. "High Error Rate" represents no constraint on accuracy during the training, "Low Error Rate" represent tightest accuracy constraints.

By treating each classification model (shown on figure 1 as "Easy", "Harder", and "Hardest") as a "black box" as opposed to a known, modifiable object, existing energy-efficient classification approaches can be directly used when constructing the system. Furthermore, multiple complex classifiers can be easily integrated into the system. Due to the modularity of our design, the system is even able to integrate humans into the loop for cases where humans have lower error rates than machines in classifying objects.

Figure 1 shows the block diagram of the proposed adaptive classifier approach. It consists of the "chooser" function (marked as ?), and several "core" classifiers with various complexities. In our case, the "chooser" function uses a multi-class logistic regression classifier, and is similar to the "Easy" classifier. The "core" classifiers shown are linear ("easy"), polynomial ("harder"), and radial-basis function (RBF, "hardest") classifiers. The linear classifier is the most energy efficient, and consists of a single multiply-accumulate block, while the RBF is the most complex, mostly due to the exponentiation function. From accuracy point of view, RBF provides the highest, while linear has the lowest accuracy. Polynomial classifier provides an intermediate point between the "easy" and "hardest". At run-time, depending on the "hardness" of data, the "chooser" function picks one of the three classifiers.

III. Evaluation

The adaptive classifier system was trained offline using MATLAB, after which the hardware design space was explored using Aladdin toolset [7]. The different design choices that we considered include level of MAC parallelization, size of SRAM for exponentiation LUT, pipeline stages, and computational parallelism. The most energy efficient system (in terms of energy-delay product) was then implemented in Chisel HDL, and the Verilog HDL source was synthesized, placed-and-routed, and extracted using 40nm Global Foundry technology and Synopsys standard cells. All simulation results are post-extraction. The datasets are provided by the UCI library [8].

Table I shows the simulation results for both adaptive and conventional classifiers. The delays represent the minimum achievable delay without change in the budgeted error rate. On average the proposed adaptive classifier approach consumes 10x more energy as compared to the linear classifier and 100x less energy than RBF. The average delay of the adaptive system across all examples is ≈2x of the linear approach, and ≈0.33x times of that of RBF. At the same time the error rate is on average 0.5% higher than RBF, but ≈ 40% lower than linear.

Figure 2 shows that by varying the error budget in the "chooser" it is possible to allocate resources to more or less complex classifier to achieve desired energy dissipation.

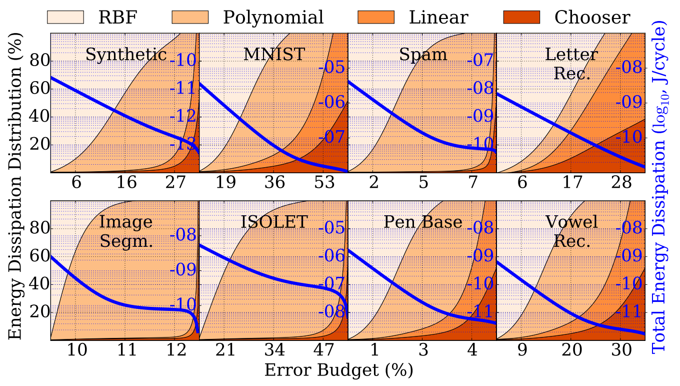

Fig. 2: Energy dissipation distribution and the total energy dissipation vs. error rate for different classification problems.

IV. Conclusion

In this work we have presented a novel adaptive classifier that can dynamically select a classifier based on the "hardness" of the current input. On average our adaptive classifier consumes 100× less energy than RBF and 10× more energy than the linear classifier. The adaptive classifier is on average ≈3× faster than RBF and ≈0.5× slower than linear approach, while the error rate of the adaptive classifier is on average only 0.5% higher than RBF but 40% lower than linear.

References

[1] N. Lane *et al.*, "A survey of mobile phone sensing," *Communications Magazine, IEEE*, vol. 48, no. 9, 2010.

[2] S. Venkataramani *et al.*, "Scalable-effort classifiers for energy-efficient machine learning," in *Proc. DAC*, 2015.

[3] P. Panda *et al.*, "Object detection using semantic decomposition for energy-efficient neural computing," *arXiv:1509.08970*, 2015.

[4] E. Park *et al.*, "Big/little deep neural network for ultra low power inference," in *Proc. CODES+ISSS*, 2015.

[5] J. Wang *et al.*, "An LP for sequential learning under budgets," in *AISTATS*, 2014.

[6] F. Nan *et al.*, "Feature-budgeted random forest," in *Proc. ICML*, 2015.

[7] Y. S. Shao *et al.*, "Aladdin: A pre-rtl, power-performance accelerator simulator..." in *ISCA*, 2014.

[8] "Machine Learning Repo," http://archive.ics.uci.edu/ml/.