Energy-Efficient Adaptive Classifier Design for Mobile Systems

Zafar Takhirov, Joseph Wang, Venkatesh Saligrama, Ajay Joshi
Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA
{zafar, joewang, srv, joshi}@bu.edu

ABSTRACT

With the continuous increase in the amount of data that needs to be processed by digital mobile systems, energy-efficient computation has become a critical design constraint for mobile systems. In this paper, we propose an adaptive classifier that leverages the wide variability in data complexity to enable energy-efficient data classification operations for mobile systems. Our approach takes advantage of varying classification "hardness" across data to dynamically allocate resources and improve energy efficiency. On average, our adaptive classifier is $\approx 100\times$ more energy efficient but has $\approx 1\%$ higher error rate than a complex radial basis function classifier and is $\approx 10\times$ less energy efficient but has $\approx 40\%$ lower error rate than a simple linear classifier across a wide range of classification data sets.

Keywords

Machine Learning, adaptive classifier, energy efficiency, mobile systems

1. INTRODUCTION

Over the last decade, there has been a shift in the consumer market towards mobile computing. Mobile workloads have become more diverse, and are increasingly trending towards utilizing computationally expensive machine learning approaches to process the large amounts of data [1]. Some examples of these problems are live translation, fingerprint matching, and diabetes testing [2]. These mobile systems however have a small form factor and run on batteries or harvest energy from the environment, and typically do not always have reliable access to the power grid. Hence, executing expensive machine learning algorithms on mobile systems with limited energy budget is very challenging. As a result, minimizing energy consumption of machine learning algorithms while achieving the desired accuracy is critical. Machine learning based classifiers have seen an increased deployment in applications such as speech recognition, handwriting recognition, as well as mobile health monitoring [3]. Here, the choice of classifier often involves a trade-off between error rate and computational efficiency. For example, accuracy-sensitive applications such as handwriting recognition have stringent error constraints, while applications like presence detection for light controls can tolerate errors. Complex classification functions such as kernel-based functions and neural networks allow for flexible boundaries to be learned yielding strong classification performance and hence can be used for accuracy-sensitive applications, but these approaches are often computationally demanding during runtime, resulting in lower energy efficiency [4]. In contrast, simple decision functions such as sparse linear functions have high energy efficiency and can be used for applications that can tolerate errors, but these approaches lack the flexibility to learn complex decision boundaries necessary for high classification accuracy.

Typically, the standard approach is to apply the same classification approach to all incoming test data during runtime. The problem with this approach is that it does not take into account the "difficulty" of the test input. For most "easy" inputs, using a complex classifier results in the same error rate as a simple classifier, but it consumes much more energy for no added benefit. Conversely, for "hard" inputs, using a simple classifier lowers energy, but it results in a lower accuracy. In this paper, we introduce an adaptive classifier approach that has a single unified "chooser" function which classifies the quality of the incoming data and then assigns it to the optimal classifier, taking both accuracy and energy dissipation into account. In our approach, the chooser function dynamically picks classifiers for each new input to maximize classification performance subject to an energy-efficiency constraint. This approach also enables one to incorporate existing as well as new machine learning approaches into constrained learning problems. Essentially, one can seamlessly incorporate optimized, complex classifiers into a budgeted system. Moreover, the budget can be dynamically changed during run-time by retraining the chooser function, which avoids the need to retrain complex classifiers. The main contributions of our work are as follows:

– We propose an adaptive classifier design that uses a regression-based chooser function to classify the incoming data as "easy", "harder" and "hardest", and based on this classification it assigns the data to the most energy-efficient classifier that meets the target error rate. This chooser function is tunable (through changing the weights of the "chooser" function) at run-time to achieve a target accuracy / error rate or a target energy efficiency.

ISLPED '16, August 08-10, 2016, San Francisco Airport, CA, USA
© 2016 ACM. ISBN 978-1-4503-4185-1/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2934583.2934615

– We implemented our adaptive classifier in hardware with Chisel [5] and Global Foundries 40nm technology node. The implementation includes the design of the chooser function and the linear, polynomial, and radial basis kernel function (RBF) classifiers. Our evaluation shows that on average our adaptive classifier consumes 100x less energy than RBF and 10x more energy than the linear classifier. The adaptive classifier is on average ≈3x faster than RBF and ≈0.5x slower than linear classifier, while the error rate of the adaptive classifier is on average only 0.5% higher than RBF but 40% lower than linear classifier.

The rest of this paper is organized as follows. In Section 2, we provide a formal definition of "hard" and "easy" problems. This is followed by a description of the adaptive classifier in Section 3 and the evaluation of our classifier in Section 4. We provide an overview of the related work in Section 5, followed by concluding remarks in Section 6.

2. HARD VS EASY CLASSIFICATION

A wide variety of classification approaches ranging from sparse linear classifiers to deep neural networks exist today, with each approach presenting a trade-off between classification accuracy and energy efficiency. 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. 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.

More formally, the behavior described earlier arises from the optimization problem when learning over training data. Given a set of n training data/label pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{1, \ldots, C\}$, where $\mathcal{X}$ is the input space, and $\{1, \ldots, C\}$ is the collection of output labels, and given a collection of k classifiers $f_1, \ldots, f_k : \mathcal{X} \to \{1, \ldots, C\}$ with associated energy budget $c_1, \ldots, c_k \in \mathbb{R}$, the goal is to learn a "chooser" function $g : \mathcal{X} \to \{1, \ldots, k\}$ that maps each example to one of the k models in input space $\mathcal{X}$ such that the average error rate is minimized subject to energy budget constraint. This optimization problem can be formulated as:

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

$$\text{Subject to: } \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} c_j \mathbb{1}_{g(x_i)=j} \leq B$$

where B is the chosen average energy dissipation budget and $\mathbb{1}_z$ is the indicator function, with a value of 1 when the logical expression z is true and a value of 0 when the logical expression z is false. By representing the constraint as a Lagrange multiplier, the problem can be represented as an unconstrained optimization problem:

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} L_j(x_i, y_i) \mathbb{1}_{g(x_i)=j}, \tag{2}$$

where $L_j(x_i, y_i) = \mathbb{1}_{f_j(x_i) \neq y_i} + \lambda c_j$ is defined as the loss associated with using the classification function f_j on example x_i, incorporating both the classification error of the prediction function as well as the energy dissipation modulated by the Lagrangian multiplier λ (in practice, the value λ is swept over in order to find a feasible system that satisfies the budget constraint[1]). This problem is equivalent to the well-studied cost-sensitive learning problem, allowing for existing learning techniques to be applied.

In practice, we upper bound the objective as an importance weighted supervised learning problem of the form

$$\min_{g \in \mathcal{G}} \sum_{i=1}^{n} W_i \mathbb{1}_{g(x_i) \neq l_i}, \tag{3}$$

where the pseudo-label l_i and importance weight W_i are defined as

$$\begin{aligned} W_i &= \max_{p,q \in \{1,\ldots,k\}} L_p(x_i, y_i) - L_q(x_i, y_i) \\ l_i &= \min_{p \in \{1,\ldots,k\}} L_p(x_i, y_i). \end{aligned} \tag{4}$$

This allows for efficient training and implementation using standard supervised learning techniques such as logistic regression and support vector machines.

The pseudo-label l_i and importance weight W_i define if an example is "easy", "harder", "hardest", etc. We consider an example to be "harder" or "hardest" if the pseudo-label l_i (representing the optimal classification system to route example i) points to a complex, energy-inefficient classifier. Conversely, we consider an example to be "easy" if the pseudo-label l_i points to an energy-efficient classifier. Note that for examples with no variation in error rate across classifiers, the importance weight W_i is generally small, as a difference in losses simply represents the difference in energy efficiency between classifiers. The over-fitting of the classifiers (including the chooser) should be addressed during the training period, however, no special steps are required to avoid such problems, as the "chooser" function is not prone to over-fitting due to the fact that it is expected to be much simpler than the most complex classifier in the ensemble. For example in our approach, the "chooser" function is a linear classifier, while the classifier for the "hardest" examples is an RBF kernel function.

The proposed system presents many beneficial properties rarely found in energy-efficient systems. By treating each classification model 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 by providing the capability to upload the data to a server to apply an extremely complex classifier. Due to the modularity of our design, the system is even able

[1]Note that we derive this problem for the energy/power constraint, however we could pose the problem as a minimum energy dissipation system given an average error constraint. Introducing a Lagrange multiplier for the average error constraint yields an identical optimization problem (modulated by a constant) as (2), and therefore solving the energy/power constrained problem is equivalent to solving the error constrained problem.

to integrate humans into the loop for cases where humans have lower error rates than machines in classifying objects.[2]

Additionally, changing the energy constraint of the proposed system does not require that the classifiers be retrained. This is particularly valuable for mobile systems, where adaptation is required to account for the difference in usage patterns of users, and energy budgets need to be updated due to changes in battery usage settings. [1]. Rather than changing the classifiers, the system only needs to retrain the chooser function g to adapt to a new budget constraint. Furthermore, the usage of a convex supervised approach such as logistic regression or a Support Vector Machine (SVM) to train the chooser function g allows for streaming online updates to be applied during run-time using stochastic gradient updates.

3. ADAPTIVE CLASSIFIERS

In this section, we present a detailed description of our adaptive classifier approach. We present the theory and the implementation details of three different classifiers and a chooser function that chooses one of the three classifiers based on the "hardness" of the data.

Figure 1 shows the microarchitecture of our proposed adaptive classifier system. It consists of the "Chooser" function (marked as $\textcircled{?}$), and several "core" classifiers with various complexities. In our case, the "chooser" function uses a multi-class logistic regression classifier. Once trained the "chooser" function selects an appropriate classifier. In the current work the "chooser" function is implemented as a logistic regression based multi-class one-vs-all classifier with three labels. Note that the adaptive classifier design proposed in this paper doesn't include the training hardware, and all the classifiers, including the "chooser" are trained offline.

During the offline training of the system, all the "core" classifiers are trained to achieve the minimum error rate possible. The "chooser" is then trained (offline) to classify the "hardness" of the inputs by analyzing the results of the "core" classifiers. For example, during the training of the "core" classifiers if a data input is classified correctly by all classifiers, it is labeled as "easy", whereas if it is classified correctly only by higher complexity classifiers, it is deemed "harder" or "hardest". The "easy", "hard" and "hardest" class labels are used to route the input to the "core" classifier of appropriate complexity. The "chooser" function is trained for a given error rate constraint, which allows the adaptive classifier system to dynamically trade off classification accuracy versus energy efficiency. The computations are performed using fixed point, 2's complement representations, thus only the most significant bit (MSB) is required to identify the label of a binary class.

Below, we describe the functional form of three "core" classification functions – linear function, polynomial function, and RBF kernels, which we used in our evaluation. The binary functional forms are presented, with multi-class prediction accomplished using a one-versus-all max-margin coding to convert from multi-class to binary classification.

We consider linear classifiers of the form

$$f_{lin}(x) = \text{sign}\left(\sum_{d=1}^{D} \alpha_d x(d) + \beta\right), \tag{5}$$

where D is the dimensionality of the data and $x(d)$ corresponds to the value of the d^{th}-dimensional element of example x. The classifier is parametrized by the weights $\alpha_1, \ldots, \alpha_D$ and the offset β, with these parameters learned using training data. The "easy" inputs are routed to this low complexity classifier. This classifier is of similar complexity as the "chooser" function, and is the simplest of the "core" functions. It is implemented as a multiply-accumulate block as shown in Figure 1(Easy). Depending on the design topology chosen after design space exploration, the multiply-and-accumulate (MAC) block is parallelized accordingly. It has the highest energy efficiency, but it also has a high error rate.

Similarly, we utilize homogeneous polynomial classifiers of the form

$$f_{poly}(x) = \text{sign}\left(\sum_{d=1}^{D}\sum_{p=1}^{P} \alpha_d^p x(d)^p + \beta\right), \tag{6}$$

where P is the power of the polynomial classifier. As in the linear case, the classifier is parametrized by the weight parameters $\alpha_1^1, \ldots, \alpha_D^P$ and the offset β. The "harder" input is routed to this higher complexity classifier block. This block is implemented as a 5^{th} order polynomial evaluation function. The "harder" block shares the multiply-accumulate unit with the "easy" block. This classifier provides a moderate compromise between error rate, energy efficiency and execution time. The polynomial is computed iteratively in five clock cycles using a MAC unit (see Figure 1(Harder)). The MAC unit is used to compute the higher order components $\alpha_\cdot^p x(\cdot)^p$ by feeding the multiplier output as the input.

Finally, we consider radial basis function kernel classifiers of the form

$$f_{rbf}(x) = \text{sign}\left(\sum_{i=1}^{n} \gamma_i e^{\frac{-\|x-x_i\|_2^2}{\sigma}}\right), \tag{7}$$

where σ is a user-specified kernel parameter and the points $x_1, \ldots, x_n$ are the training data points. The classifier is parametrized by the kernel weights $\gamma_1, \ldots, \gamma_n$. The "hardest" input is routed to this highest complexity classifier block.

[2]In this case, a second budget constraint may exist regarding the number of times a human is queried. Due to lack of space, we do not discuss this model here, however the exact same reduction to a cost-sensitive learning problem can be made by incorporating a second Lagrange multiplier for this new constraint.

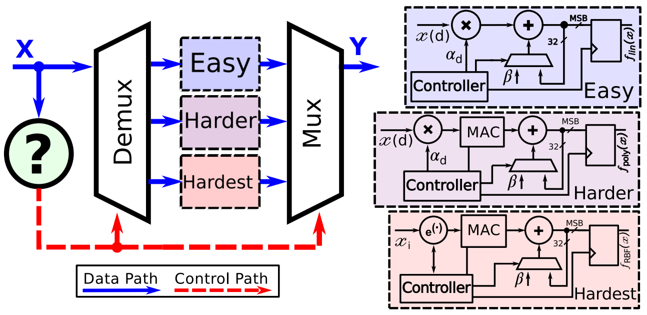

Figure 1: Conceptual block diagram of the proposed approach. "Chooser" function is marked as $\textcircled{?}$, and its microarchitecture is the same as the "Easy" block. Data paths are shown as solid lines, while control path is dashed.

Because RBF includes an exponential function, this block is more complex compared to the previous two. An SRAM is used as a look-up table (LUT) in order to avoid lengthy computations of the exponential function. The RBF function computation partially reuses the blocks in the other two classifiers in order to reduce area and increase hardware utilization. The block diagram shown in Figure 1(Hardest) includes an exponential function that is implemented as a look-up table. The MAC unit then computes $\gamma_i e^{(\cdot)}$.

4. EVALUATION

In this section we provide a detailed evaluation of the proposed adaptive classifier.

4.1 Experimental Setup

Before exploring the hardware design space of our adaptive classifier, the training of the adaptive classifier system was performed offline using MATLAB for several different constraints scenarios, where we set the maximum acceptable error rate independent of the application. Our goal was to minimize the energy dissipation for a given error rate. In MATLAB, the minimum energy dissipation point is equivalent to the use of the simplest[3] model possible for the given constraints. The "chooser" function was trained to achieve minimum energy dissipation by optimizing the subblock utilization for a given error rate, i.e. if two classifiers with different energy dissipations have comparable error rates, the "chooser" function would be biased towards the simpler classifier. The data sets were separated into training and evaluation sets using an 80/20 scheme, where 80% of the inputs were used for training, and 20% for evaluation. All figures of merit in the evaluation section were acquired using the "evaluation" data sets, thus guaranteeing that inputs were never seen before by the systems under test.

After the training, in order to explore and narrow down the hardware design space of our adaptive classifier, we used the Aladdin toolset [7]. The different design choices that we considered include level of MAC parallelization, size of SRAM for lookup table, level of computational parallelism, and the number of pipeline stages. Figure 2 shows the results obtained using the Aladdin tool for one of the input data sets (`Letter Recognition`). Here, all the EDP points

[3]Simplest model means the one that has the smallest number of computations per example. For example, if a linear and a polynomial function give the same error rate for a given input, the linear function is considered to be the simplest.

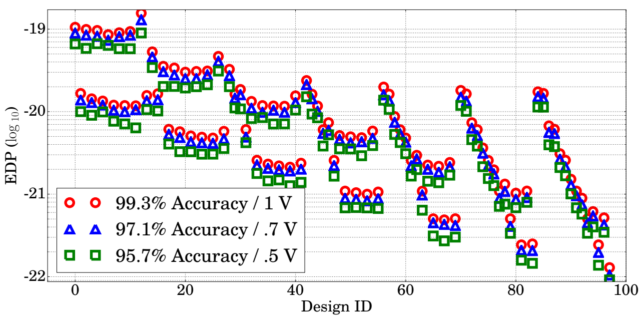

Figure 2: Energy delay products for different adaptive classifier designs generated by Aladdin [7].

are computed for the maximum statistical classifier accuracy and are acquired by varying the size of the computational units, memory size, level of computational parallelism, and number of pipeline stages across the different designs. The scatter plot shows how the various design choices affect the EDP of the system. The figure also shows the EDP at different voltages and the corresponding accuracy levels. The goal for the current work was minimizing the EDP for the given error rate constraint. Hence we selected the designs with the minimum EDP.

The chosen design was then implemented using Chisel [5]. This environment was chosen, as it generates both hardware description code (Verilog) as well as C++ functional model. That allows us to perform a rapid verification of the design. The Verilog HDL code was then synthesized and placed-and-routed using 40nm Global Foundries technology with Synopsys standard cell libraries. The delay and energy dissipation were acquired from placed-and-routed, RC-extracted layout simulations using Cadence toolset. The main figures of merit that were considered for evaluation were computational performance (defined as average computation time per input), energy dissipation, and statistical performance, defined as misclassification error rate. We used seven different data sets from the UCI library [8] and a synthetic data set for our evaluation. The list of the used data sets is shown on Table 1 under the "Data Set" column. In addition to the adaptive system, individual classifiers were implemented for comparison and validation.

4.2 Experimental Results

Figure 3 shows the Error Rate vs. EDP of the proposed approach and that of the linear, polynomial, and RBF classifier when used individually. We can see from the figure that depending on the input data set the proposed approach behaves differently. Adaptive classifier system performs best when the input data set has uniformly distributed "hardness", meaning the input has an equal mix of "easy", "harder" and "hardest" examples. If the input data set is mostly "easy" or mostly "hard", adaptive classifier would waste energy because a simple or complex classifier could be used, respectively, without the "chooser" function. As seen from the data plots, linear classifier cannot achieve low error rates, while RBF, although showing extremely low error rates, has high EDP. For example, `Letter recognition` data set shown in Figure 3 shows that linear classifier cannot achieve an error rate below 65%, while RBF, which has error rate close to 0%, has very high delay and energy dissipation. Thus, it is important to analyze the input data set and application before implementing the "chooser" function.

Unlike the conventional classifiers, the adaptive classifier system enables us to trade-off error rate and EDP. For example, in case of the `Letter Recognition` data set, if the EDP constraint falls between the EDP of the linear and polynomial classifiers, then in the conventional approach one has to choose the linear classifier which results in a high error rate. However, if one were to use an adaptive classifier, then the "chooser" function could be tuned to the EDP constraint and an error rate lower than that of a linear classifier can be achieved. Similar approach can be adopted for EDP constraints lying between polynomial and RBF classifier for the `Letter Recognition` data set as well as for other data sets to achieve lower error rates compared to conventional classifiers.

	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 1: Energy and Delay for the adaptive classifier and conventional classifiers for different data sets. Each row corresponds to subplots in Figure 3. Energy results are shown in J/cycle and delay is represented as μs.

	Current Work		SEC	
	Time	EDP	Time	EDP
Synthetic	1.67e-04	1.81e-19	1.72e-04	3.86e-19
Image Segm.	2.60e-03	1.30e-17	3.22e-03	3.99e-17
ISOLET	2.51e-02	8.98e-12	2.50e-02	1.78e-11
Letter Rec.	4.86e-03	3.27e-17	5.42e-03	8.13e-17
MNIST w/ bkgnd	3.69e-02	1.77e-11	3.67e-02	3.48e-11
Penbase Rec.	5.15e-03	3.23e-17	5.57e-03	7.54e-17
Spam	2.83e-03	4.73e-15	3.39e-03	1.36e-14
Vowel Rec.	4.03e-03	1.91e-17	4.05e-03	3.85e-17

Table 2: Comparison of the proposed and SEC approaches [6] trained for low energy subject to error rate budget. The time is in seconds.

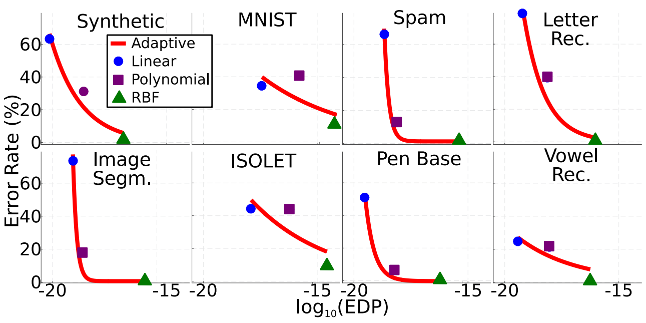

Figure 3: Error rate vs. Energy Delay Product (EDP) for adaptive classifier and conventional classifiers. Adaptive classifier can be tuned to achieve any of the intermediate error rate values.

Table 1 shows the energy dissipation and delay values of the adaptive classifier when running the various data sets. The delays represent the minimum achievable delay without change in the budgeted error rate. “High error rate” represents the training scheme where we do not constrain the error rate (due to which the linear classifier is always preferred) resulting in lowest EDP. On the other hand, the “Low error rate” column shows the training scenario where the target error rate budget is set to be low (due to which mostly RBF is selected by the “chooser” block) resulting in a higher EDP. 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.

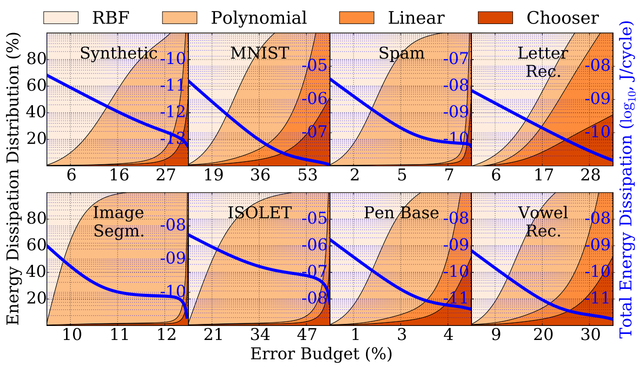

Figure 4: Energy dissipation distribution and the total energy dissipation vs. error rate for different classification problems.

From Table 1 we can see that the “High Error Rate” and “Low Error Rate” results (both energy and delay) are comparable to linear and RBF classifiers respectively. This behavior is expected, because if the training is biased towards one of the error rate extremes, the adaptive classifier tends to choose only linear or RBF. This is also confirmed by the subsystem energy distribution plots shown on Figure 4. This figure shows the total energy consumed by the adaptive classifier system at different error budgets as well as the distribution of energy consumption across different parts of the system. Note that the “chooser” function tends to use linear classifier more often when a high error rate is acceptable, while being biased towards using RBF when the error rate constraints are tight. That means that energy dissipation of the linear classifier dominates at high error rate budgets. This allows for dynamic change in energy dissipation depending on the current error rate budgets as well as complexity of the current computation.

Figure 5 shows the area results post-placement-and-routing. The bars show the relative areas of different blocks, while the solid line shows the total area. The results include the SRAM LUT as part of the RBF block. This LUT occupies 43900 μm^2 and is used to evaluate the exponentiation function. Note that the area distributions are different across different data sets as we generated a unique classifier design for every data set separately. If we were to design the adaptive classifier using all datasets together, then it would lead to a sub-optimal design.

As part of our evaluation, we have performed a detailed comparison of our approach to the state-of-the-art scalable

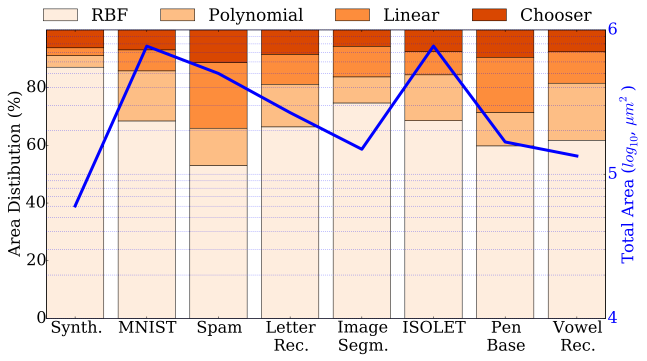

Figure 5: Relative area distribution of subblocks in adaptive classifier (bars) as well as the total area (solid line).

effort classifier (SEC) approach described in [6] (see Table 2). To make the comparison as fair as possible, SEC system was designed using the same process as the proposed approach. We used our linear, polynomial, and RBF classifiers to implement the three stages of the SEC, and the SEC design space exploration was performed using Aladdin tool. However, the energy results for the SEC approach were from just synthesized designs, while the adaptive classifier approach energy results were using netlists extracted from placed-and-routed designs. The "Time" column shown in Table 2 shows the results of C++ simulations. Results show that the execution time of for both approaches is comparable. However, the adaptive classifiers can achieve almost 2× lower EDP while operating at the same error rate. The reason for this is that when facing "harder" problems, the SEC approach still utilizes "easy" stages of the classification system before switching to more complex classifiers, thus wasting energy. Adaptive classifier approach chooses only one (appropriate) classifier depending on the "hardness".

5. RELATED WORK

There has been some prior work in the design of energy-efficient classifiers. The system proposed by Venkataramani et al. [6] 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. [9] introduce a system for object detection using energy-efficient neural computing. The hierarchical framework of classifiers were 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. [10] uses dynamic threshold adjustment in deep neural networks (DNN) to achieve low power operation. In this work, a "little" DNN is used, and if it fails, a "big" DNN is used, etc. Because of the system topology it is not possible to automatically enable the necessary DNN without running all other DNNs.

The approach proposed in the current work could be viewed as the energy-efficient learning problem, which in turn is a variation of the problem of learning under test-time budgets [11, 12, 13, 14, 15, 16]. Here, learning under test-time budgets is a family of approaches that focus on minimizing the costs during testing. In general, these approaches focus on selecting sensors as opposed to energy expenditure and ignore the energy usage associated with selection system. As the main obstacle in the test-time budget problem is the sequential revelation of information, the energy expenditure of the selection system is generally not accounted for. Additionally, many approaches to test-time budgeted learning [14, 15, 16] require design of both the resource allocation as well as the classification functions, preventing the use of optimized modular systems.

6. CONCLUSION

In this work we have presented a novel and fundamentally different classifier approach that can dynamically trade-off error rate for energy dissipation while solving a classification problem. Our adaptive classifier design uses a logistic regression classifier as its "chooser" function to identify the input data complexity and then route the data to the classifier with appropriate complexity. We implemented (placed-and-routed design) the adaptive classifier system using a 40 nm technology, and tested it using the data set provided by the UCI repository. Our evaluation shows that on average our adaptive classifier consumes 100x less energy than RBF and 10x more energy than the linear classifier. The adaptive classifier is on average ≈3x faster than RBF and ≈0.5x 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.

7. REFERENCES

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

[2] K. Sowjanya *et al.*, "Mobdbtest: A machine learning based system for predicting diabetes risk using mobile devices," in *Advance Computing Conference (IACC)*, 2015.

[3] G. Wu *et al.*, "Always connected: Billions of connected nanosystems," in *Workshop on Rebooting the IT Revolution*, 2015.

[4] S. Venkataramani *et al.*, "Approximate computing and the quest for computing efficiency," in *Proc. DAC*. ACM, 2015.

[5] "Chisel: Constructing Hardware in a Scala Embedded Language," https://chisel.eecs.berkeley.edu/.

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

[7] Y. S. Shao *et al.*, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA*, 2014.

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

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

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

[11] K. Trapeznikov *et al.*, "Supervised sequential classification under budget constraints," in *AISTATS*, 2013.

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

[13] ——, "Model selection by linear programming," in *European Conference on Computer Vision, 2014*.

[14] Z. Xu *et al.*, "Cost-sensitive tree of classifiers," in *Proc. ICML*, 2013.

[15] M. Kusner *et al.*, "Feature-cost sensitive learning with submodular trees of classifiers," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

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