2018 IEEE 27th Asian Test Symposium

# RiskCap: Minimizing Effort of Error Regulation for Approximate Computing

Shuhao Jiang, Jiajun Li, Xin He†, Guihai Yan, Xuan Zhang†, Xiaowei Li
State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
†Washington University in St. Louis
{jiangshuhao, lijiajun, yan, lxw }@ict.ac.cn, {xin.he, xuan.zhang}@wustl.edu

***Abstract*—Quality management, which is responsible for controlling approximation quality to meet user requirement, plays a key role in the applicability of approximate computing. An effective and efficient quality management needs to be accurate to detect intolerable errors meanwhile light-weight in nature. However, it is difficult to design such a quality management satisfying both the two demands and existing work usually optimizes for one demand at the expense of the other. In this paper, we aim to achieve higher energy efficiency of quality management by optimizing detection accuracy and overhead simultaneously. We observe that the detection difficulty varies across inputs and there exists much redundant computation in detection process. Based on this observation, a cascaded quality management which can minimize the overhead and doesn't lower detection accuracy is proposed. The proposed solution pays more proper computation effort according to different detection difficulties of inputs so as to avoid unnecessary energy consumption. What's more, by exploring the design space sufficiently and effectively, we can assure the highest energy-efficiency of the proposed topology. The experiment results demonstrate that our approach can achieve much greater energy-efficiency than existing solutions.**

***Index Terms*—approximate computing, quality management, progressive detection**

## I. Introduction

With the staggering Moore's law and Dennard scaling [1], approximate computing is believed to be a promising computing paradigm to improve energy efficiency, especially for those applications with intrinsic error resilience [2]. Surprisingly, "resilient" applications have been found in a broad range of domains such as machine learning, computer vision, web search, and cyber-physical systems [3]–[5].

Many approximation techniques have been proposed, including approximate LUT [5], [6], approximate logic and memory [7] and approximate accelerators [4], [8]–[11]. Although these techniques fuel better energy efficiency, how to guarantee the quality and acceptability of final results is still an open problem. An effective computing quality management scheme, which plays a decisive role in the applicability of approximate computing, is critical.

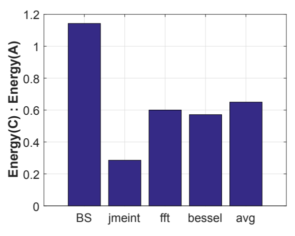
The basic task of quality management is to detect unacceptable errors (i.e. quality violations) and recover them for satisfying certain level of target quality requirement. Rumba [12] proposed three light-weight predictors to predict quality violations based on input data. However, these predictors turn out to be too simple to deliver good enough prediction accuracy. Low prediction accuracy implies more false positives or false negatives. False positive would lead to unnecessary recoveries thus dampening the energy benefit from approximate computing while false negative would incur the risk of unacceptable quality degradation.

To achieve higher accuracy, more sophisticated quality managements are proposed. Wang et al. [13] proposed a quality management design consisting of several light-weight predictors. Similarly, MITHRA [14] proposed a neural network based detector. However, the starting point of these work is mainly for enhancing accuracy so such methods induce considerable overhead. High overhead of complicated predictors offsets a considerable proportion of benefits from approximate computing, and results in the limitation of the overall energy enhancement. For example, the proposed solution in [13] only achieve 11% to 23% energy savings over simple predictors of Rumba.

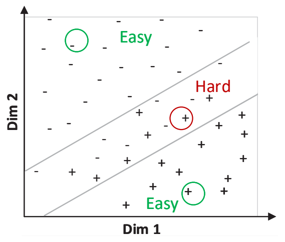
This problem has also been noticed by some researchers. For example, Xu et al. [15] tried to obtain higher energy-efficiency through approximate accelerator and error controller (for quality manangement) co-design. Nevertheless, the inefficiency problem caused by overhead of error controller is still severe. Fig. 1(a) compares energy of approximate accelerator (A) and error controller (C) in [15]. On average, the energy of error controller occupies up to 65% of the accelerator.

Due to the great difficulty of traditional tradeoff between high accuracy and low overhead, more new beneficial properties should be explored to improve energy efficiency of quality management. One of promising properties is that the detecting difficulty of quality violations varies widely across input data. This phenomenon is quite intuitive. As shown in Fig. 1(b), suppose a simple case including 2-D input data and corresponding distributions of quality violation (label " $+$ ") and safe approximation (label " $-$ "), then it is easy to find that the points in centering band is harder to detect than those out of band. This observation implies that computation effort of error controlling should be judiciously allocated according to the difficulty of input instances. However, existing work on quality management expends equal effort on all inputs and lacks effective methodology to mine this potential benefit.

In this paper, we propose RiskCap, an efficient quality management which is optimized on both detection accuracy and overhead. RiskCap is a cascaded structure consisting of



(a)



(b)

Fig. 1. (a) Energy comparison in [15]. (b) Variety of detection difficulty.

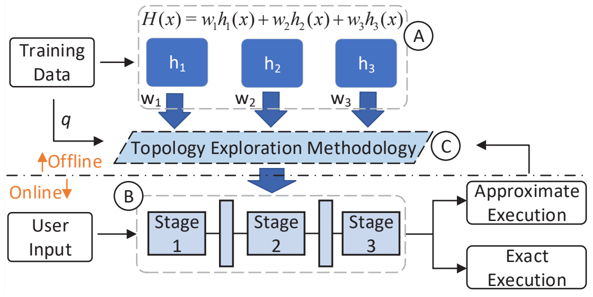


Fig. 2. The framework of RiskCap.

multiple stages. The key of this cascade design is to realize such a procedure: The easy-to-detect inputs can terminate at earlier stages while hard-to-detect inputs will go deeper to be analyzed further. We call this procedure as *progressive detection*. Through progressive detection, RiskCap only takes necessary effort according to detection difficulties of inputs, thus reducing the overhead of quality management. What's more, the overhead reduction is not at cost of detection accuracy drop. Besides, we also design a low-overhead cascaded structure to support progressive detection. At last, an efficient heuristic algorithm is presented to explore the optimal topology configuration (e.g. number of stages, size of each stage etc.) for highest energy-efficiency. Overall, RiskCap can achieve higher energy-efficiency owing to high detection accuracy and efficient progressive detection. The experiment demonstrates that our approach can achieve much higher energy-efficiency than existing solutions.

The rest of the paper is organized as follows. In Section II, we gives an overview of related work. In Section III, we introduce the design and structure of RiskCap. Experimental results are given in section IV. At last, section V concludes the paper.

## II. Related Work

The basic idea of approximate computing is to simplify the exact computing or replacing it with approximate versions to achieve energy-efficiency gains. Although errors are introduced to outputs in this process, they will still be acceptable if the quality loss of outputs is under the tolerant range. Over the years, many approximate computing techniques have been proposed. Loop perforation [3] skips the iterations of loops randomly to reduce computation. Approximate accelerators [4], [8], [16] are designed to replace the exact execution of kernel code regions to speed up error-resilient applications. Accuracy-configurable logic units like [17] are proposed to realize approximate computing on circuit level.

However, the availability of approximate computing techniques depends heavily on whether quality of approximate output satisfies the requirement. The quality requirements usually are some defined metrics, e.g. PSNR for image processing applications. To ensure output quality satisfies the requirement, quality management should detect unacceptable errors(i.e. quality violations) effectively. Green [18] and SAGE [19] runtime check the output quality periodically by comparing sampling approximation results and accurate results. But these methods failed to cover the large-error outputs that escape the sampling while checking all outputs is too computation-expensive.

Using machine learning methods to predict quality violations based on input data of applications turn out to be a promising technique to quality management. Rumba [12] proposed 3 kinds of light-weight predictors. By checking errors according to inputs, Rumba can cover all data meanwhile consuming little energy. But the three predictors are based on simple models, so the prediction accuracy is limited.

Therefore, more complex predictors are presented for higher accuracy. Ting Wang et al. [13] combine weak predictors to a strong predictor for higher accuracy. Mahajan et al. [14] proposed a neural-based predictor with statistical guarantees. But the starting point of these work is simply for enhancing accuracy so considerable overhead is brought, as a result, the overall energy enhancement is limited.

To light the over-weighted predictor, Xu et al. [15] propose an iterative training method to explore the topologies of the accelerator (which is approximate) and the predictor comprehensively. Therefore, a more balanced topology of approximate accelerator-predictor framework is achieved for higher energy-efficiency. But the optimal topology shown in [13] indicates that the predictor size is almost equal to the approximate accelerator, so more novel and effective methods to lighten the predictor is desired.

Utilizing variety of classification difficulty to reduce energy of machine learning algorithm is an active area of research. Recent work [20], [21] has also proved that this property exists pervasively in many other applications and achieved considerable improvement on energy and performance through multiple classifiers. However, these work tried to transform the indivisible classifiers into multiple components, which will bring considerable overhead. In addition, it lacks theoretical foundation to prove this transformation is trustworthy enough to maintain the original classification capability, which is critical to quality management.

We propose a cascaded structure for minimizing effort of quality management by progressive detection. Unlike existing work, which pays equal detection effort on all invocations, our methodology can dynamically adjust computation effort according to inputs. Besides, since the proposed structure is cascaded in nature (described in section III), we don't need to pay much overhead for transformation and worry about the degradation of detection accuracy.

## III. The Framework of RiskCap

The framework of RiskCap is shown in Fig. 2. The procdedure of RiskCap is divided into two phases, offline phase and online phase. In offline phase, we train the cascaded structure according to representative training data, quality requirement and given approximate method. In online phase, we detect each user input should be executed in approximate version or exact version.

Besides, the framework of RiskCap consists of three components. The first component Ⓐ is an ensemble learning model [22] on which progressive detection is based. Ensemble methods combine several weak (light weight but low accuracy) detectors to achieve higher detection accuracy. In ensemble learning, the weak detector is known as basic detector. A typical combination method is to combine the detection outputs of basic detectors in a weighted averaging way, which makes ensemble models cascaded in nature. In Fig. 2, outputs of basic detectors are represented by $h_i(x)$ and their combination is $H(x)$. The second component Ⓑ is the cascaded structure that supports progressive detection. The last component Ⓒ is the heuristic algorithm to find the optimal topology on energy-efficiency.

We will explain how the progressive detection (based on the ensemble model) works in section III-A. The low-overhead cascaded structure design is presented in section III-B. The heuristic algorithm to find the optimal topology is described in section III-C.

### A. The Progressive Detection

The progressive detection is to terminate early for easy-to-detect inputs while analyze further for hard-to-detect inputs. We construct the progressive detection based on two basic observations: (1) A strong (high accuracy) detector can be constructed through combining several weak detectors; (2) The quality management in approximate computing can be regarded as a binary classification task. The first observation is validated by ensemble learning methods like adaboost and random forest algorithm. A typical prediction methodology of ensemble learning is as follows:

$$H(\boldsymbol{x}) = \sum_{i=1}^{N} w_i h_i(\boldsymbol{x}) \tag{1}$$

where $H(\boldsymbol{x})$ is combined detection result, $h(\boldsymbol{x})$ and $w$ denote detection result and weight value of individual basic detector respectively, and $N$ is the number of basic detectors. Basic detectors have similar structure but different parameters. Based on the first observation, although basic detectors yield limited accuracy because of their lightness, the combination of them can yield high detection accuracy. Note that all basic detectors have indispensable contribution to final result so none of detectors is redundant. This observation guarantees the feasibility of cascaded structure with high detection accuracy, and the simplest example is that we can take each basic learner as one stage.

However, Eq. 1 doesn't guarantee the progressive detection because the combined result is fluctuant in common scenarios,

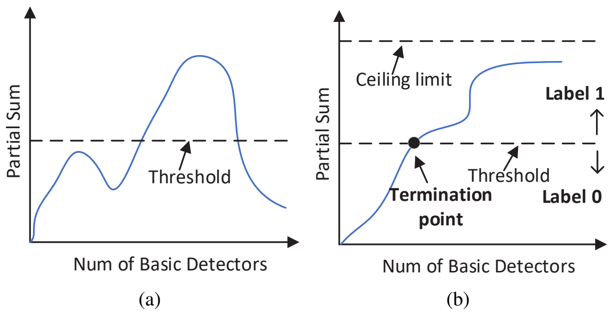


Fig. 3. (a) Combination procedure of regression. (b) Combination procedure of binary classification.

such as regression. Fig. 3(a) shows the conceptual view of this scenario. In a regression task, although weights $w$ are all positive, $h(x)$ can be positive and negative, so the final result can not be determined until the last detector finishes. Obviously, progressive detection is infeasible in such scenarios. To enable progressive detection, the combination result must be an incremental procedure, and a typical example is the binary classification. As shown in Fig. 3(b), in binary classification, $h(x)$ is either 0 or 1, so the combination result keeps growing from beginning to end. Once the partial sum exceeds the threshold (typically 1/2 of ceiling limit), the computation can be terminated safely without worrying about degradation of accuracy. As the second observation indicates, the quality management in approximate computing can be abstracted as a binary classification task [14], where class 1 donates a quality violation while class 0 means a safe approximation. As a result, the two observations ensure progressive detection with high accuracy.

### B. Cascaded Structure Design

We construct the cascaded structure of RiskCap with two main components, *stage* and *valve*. Fig. 4 shows a 3-stage case of the design. The stage component includes one or more basic detectors to detect quality violations. The basic detectors can be *linear model*, *decision tree* and even shallow *neural network* depending on the difficulty of quality controlling. In train phase, we explore the optimal stage topology for energy-efficiency such as the number, size and execution order of basic detectors and number of detectors in each stage, and detailed algorithm will be presented in section III-C.

The valve component is used for progressive detection at runtime. The logic is as follows: When a binary $h(\boldsymbol{x})$ is calculated by one stage, we add product of $w$ and $h(\boldsymbol{x})$ to the current partial sum (P sum/N sum). Then we compare the new partial sum with threshold (i.e., 0.5 when the ceiling limit is 1). If the partial sum exceeds threshold, which means the detection result has been able to be determined now, we turn off the valve so the subsequent stages will not be activated for current input (e.g. stage 3 in Fig. 4). P sum and N sum represent the positive partial sum (for class 1, i.e. quality violation) and negative partial sum (for class 0, i.e. safe approximation) respectively, in other words, we check the possibility of early termination for both classes, and condition triggering of either class will turn the valve off. Because the

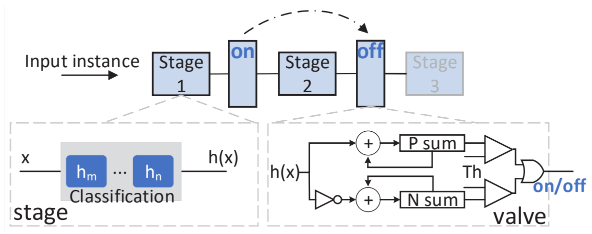


Fig. 4. The cascaded structure of RiskCap.

valves between stages don't execute simultaneously, the valve component is reused through whole progressive procedure so the overhead is negligible.

### C. Topology Exploration Methodology

The goal of topology exploration is to find most beneficial structure for energy-efficiency in offline phase. Various stage topologies correspond to different accuracy-overhead tradeoff points in feasible solution space. We conclude that determinative knobs of stage topology include the *size*, *number* and *execution order* of basic detectors and how to *map* detectors to stages. The effect of *size* and *number* knobs is intuitive. Take decision tree as an example, *size* means number of nodes in one tree and *number* represents number of trees. High value of *size* and *number* can yield high detecting accuracy while inducing high overhead. Order of detectors is also a critical factor for energy-efficiency. Because of progressive detection, easy-to-detect inputs will terminate earlier, but the termination location may heavily depend on the execution order of basic detectors. Considering the inevitable diversity in accuracy of detectors, it may not be a wise choice to take the "worst-behavior" detector to execute first. At last, how to map detectors to stages is also an essential problem. The case that one basic detector maps one stage will lead to maximum invocations of valve while mapping multiple detectors to one stage may miss opportunities of early termination. We find it's more important for energy-efficiency to maximize opportunity of early termination from experiments, so one to one mapping is better.

As shown in Algo. 1, we propose an efficient heuristic algorithm to find the most energy-efficient topology because exhaustive searching in all topologies is very time consuming. Firstly, we estimate the energy consumption of original approximation case which has no quality management (line 1). Then we reduce search space of *number* and *size* by setting quality and energy limits. Since the accuracy increase is marginal with adding the number of detectors [22], we predefine a conservative value of *number* where accuracy increase has stopped (line 3). Then (to get smallest size $BT_{low}$) we iteratively decrease *size* until approximation error of quality management violates the target requirement $q$ (line 4-7). Similarly, we determined biggest *size* ($BT_{high}$) with which energy of quality management doesn't exceed certain proportion ($\gamma$) of original energy consumption (line 8-11). Candidate topologies are constructed based on traditional training procedure of ensemble learning methods, such as adaboost, random forests (line 12). After that, we determine the optimal execution order by developing two critical metrics(line 13):

**Algorithm 1** Methodology to explore the optimal topology

```
Input: Basic detector set $BT$, training data $D$, approximate method $A$, quality requirement $q$
Output: Optimal stage topology $T_{opt}$
1: $E_{orig} = Energy(A, D)$
2: Initiate $\gamma$
3: $N = SetMaxNum()$
4: while $Error(A, BT, N, D) < q$ do
5:     $BT = Decrease(BT)$
6: end while
7: $BT_{low} = BT$
8: while $Energy(BT) < \gamma * E_{orig}$ do
9:     $BT = Increase(BT)$
10: end while
11: $BT_{high} = BT$
12: $CT = Train([1, N], [BT_{low}, BT_{high}, D])$
13: $[c\_vec, w\_vec] = GetPara(CT, A, D)$
14: $metric\_vec = c\_vec. * w\_vec$
15: $prioriy = Sort(metric\_vec)$
16: $T_{opt} = CT[prioriy]$
17: return $T_{opt}$
```

- *Coverage* of a basic detector ($c_{vec}$ in Algo. 1), which indicates what proportion of input instances can be classified accurately on train data. Larger the coverage is, the more likely input instances terminate early.
- *Weight* of a basic detector ($w_{vec}$ in Algo. 1). Obviously, the weight $w$ reflects the influence of a basic detector on final results. Basic detector with larger weight should be granted higher priority.

In detail, $w_{vec}$ can be achieved directly from traditional training procedure and we achieve $c_{vec}$ by calculating prediction accuracy of each trained detector on training data.

Using the two metrics, we avoid to evaluate all the execution order cases thus reducing the exploration space further. Then we reorder the computation priority of candidate topology (line 14-16) and get the most energy-efficient topology.

## IV. Evaluation

### A. Experimental Setup

Similar to existing work [12]–[14], we evaluate our proposed framework with Neural Processing Unit (NPU) from [4], in which a neural network is trained to approximate the execution of computation-intensive code. Then we use RiskCap to control the quality of NPU approximation at different quality requirements. In detail, we construct RiskCap by utilizing decision tree as the basic detector and adaboost for training. The cascaded structure and topology exploration is given in section III-B and III-C. Then we estimate the efficiency of proposed framework to manage the quality of NPU approximation at different levels. *It should be noticed that the proposed scheme is not limited to NPU but suitable for other approximate techniques as well.*

**Application benchmarks:** We select 5 benchmark applications from [4], as shown in Table I. The NPU topologies, train data and test data are listed. The last column of Table I is the evaluation metric on approximation quality and the initial error values which are measured by always invoking approximate

TABLE I
BENCHMARK INFORMATION AND ACCELERATOR TOPOLOGIES

| Applications | Domain | Train Data | Test Data | NN Topology | Evaluation Metric |
|---|---|---|---|---|---|
| **BS** | Financial Analysis | 100k inputs | 200k inputs | 6->8->8->1 | Mean Relative Error(14.90%) |
| **inversek2j** | Robotics | 50k random (x,y) point) | 500k random (x,y) point | 2->2->2->2 | Mean Relative Error(10.45%) |
| **jmeint** | 3D Gaming | 10k pairs of 3D triangles | 1000k pairs of 3D triangles | 18->32->8->2 | Miss Rate(20.72%) |
| **kmeans** | Machine Learning | 50k pairs random (r,g,b) values | 512x512 pixel image | 6->8->4->1 | Mean Output Diff(8.40%) |
| **sobel** | Image Processing | 512x512 pixel image | 3x512x512 pixel images | 9->8->1 | Mean Pixel Diff(15.8%) |

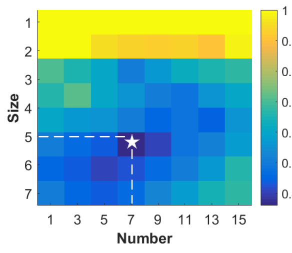
NN accelerators. For quality management, we choose decision tree as the basic detector to construct cascaded controller and combine basic detectors in adaboost algorithm [22].

**Energy modeling:** We collect the architectural activities of the NN accelerator using GEM5 [23] simulator. After that, McPAT [24] is employed to estimate energy consumption. The micro-architectural configurations of X86_64 CPU and NN accelerators parameters for GEM5 are consistent with [12]. We calculate RiskCap quality management separately and specify it at the register-transfer logic (RTL) level, then we use Synopsys Power Compiler to estimate its energy consumption.

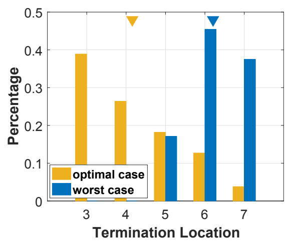
## B. The Effectiveness of RiskCap

In this section, we present experimental results to demonstrate the effectiveness of topology exploration and progressive detection of RiskCap.

*1) Topology Exploration:* Fig. 5(a) shows the reduced space of topology exploration on benchmark BS, by utilizing the algorithm presented in section III-C. The topology exploration on other benchmarks follows the same methodology. The x-axis of Fig. 5(a) denotes the *number* of detectors, we predefined this value as 15, which is a empirical value based on enough experimental trials, and higher value turn out to have quite limited enhancement on accuracy. The y-axis is the scope of detector *size*, which is determined with quality and energy constraints. From 35 candidate topologies, the optimal one is stared and it's *number* and *size* are 7 and 5 respectively. After that, we decide the optimal *execution order* based on metric *coverage* and metric *weight*. To illustrate the impact of execution order on early termination and the effectiveness of our metrics, Fig. 5(b) compares the worst order and optimal order case on distribution of early termination. By tracing termination locations, we can conclude that the distribution of termination locations in optimal order case is more biased to the left than that in worst case. Overall, average termination location in optimal order is 4.13 while the value in worst case is 6.20, which increases 30% additional computation. When the optimal order of each candidate topology is determined, the



(a)



(b)

Fig. 5. (a) Topology exploration in benchmark BlackScholse (BS). (b) The impact of execution order on early termination in benchmark BS.

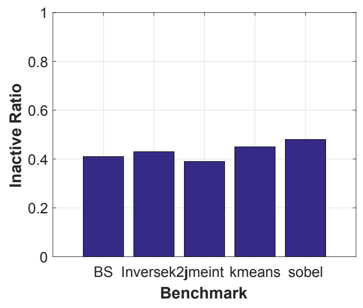


Fig. 6. Inactive ratios in different applications.

most energy-efficient one can be found easily. For benchmark BS, the optimal topology is marked in Fig. 5(a).

*2) Inactive Ratio:* To demonstrate the potential benefits of progressive detection, we define the *inactive ratio* as a metric. Since progressive detection can realize early termination of detection, the subsequent detectors after termination location will be inactive state. Inactive ratio denotes the ratio of average number of inactive detectors compared with number of all detectors. As shown in Fig. 6, the inactive ratio is above 40% in almost all applications, which implies we can save considerable computation by progressive detection.

## C. Prediction Accuracy

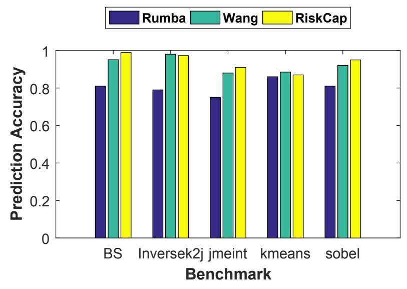
The prediction results are presented in Fig. 7 when targeting on 7.5%, 5.0% and 2.5% quality loss. We also present the best prediction accuracy of detectors in Rumba [12] and Wang's work [13] for comparison.

According to the results shown in Fig. 7, we demontrate that the proposed framework can yield high enough detection accuracy compared with existing work. For example, targeting on 7.5% quality loss(i.e. 92.5% quality requirement), Rumba can only achieve 81% accuracy on *sobel* benchmark while the prediction accuracy of our framework is 97%. Higher accuracy indicates that more computation can be executed in approximate version without violating the quality requirement thus bringing more energy benefit. Besides, Wang's work can also achieve 95% accuracy, which is comparable with our results. However, considering the limited space of further enhancement on accuracy and the high overhead of Wang's design, it is an acceptable result.

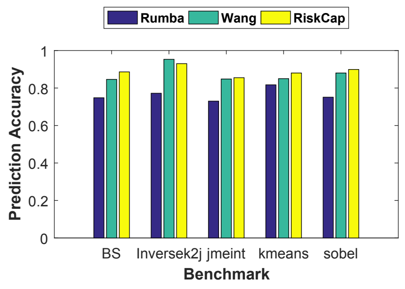
## D. Energy Savings

In this section, we compare the system energy efficiency of different quality detectors. Fig. 8 shows the energy consumption using different quality detectors when targeting on 5.0% quality loss requirement.

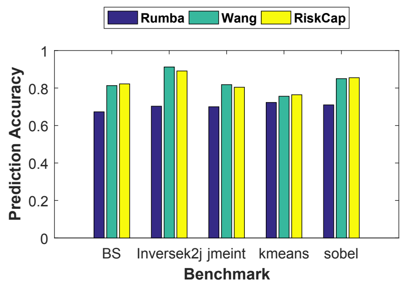
Through the results shown in Fig. 8, we find that our quality management can always achieve lowest energy consumption.



(a) Target 7.5% quality loss



(b) Target 5% quality loss



(c) Target 2.5% quality loss

Fig. 7. The prediction accuracy when targeting different error constraints.

Take the *sobel* as an example, the proposed detector can achieve 42.1% energy reduction over Rumba, whose energy is normalized to 1 as the baseline. Compared with Wang's detector, our detector can achieve 20.0% energy reduction on *sobel*. The energy benefit of RiskCap comes from two reasons: the first one is high prediction accuracy, which reduces the energy consumption of unnecessary recoveries. And the second one is progressive detection with optimal topology, which can dynamically adjust computation effort based on controlling difficulty other than paying equal effort on all inputs, thus lightening the overhead of quality controlling greatly. Because of lacking effective methodology to achieve both properties, existing work can not provide comparable energy efficiency compared with our design. On average, RiskCap can achieve 25.4% energy benefit over Wang and 40.6% energy benefit over Rumba.

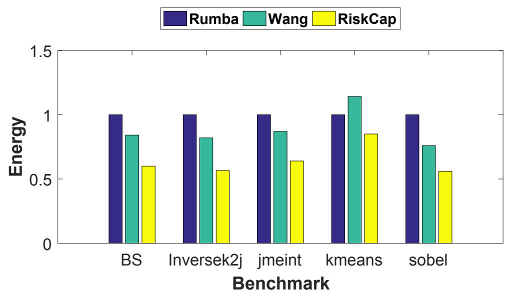


Fig. 8. The normalized energy of different predictors (5.0% quality loss).

## V. Conclusions

In this paper, we proposed a cascaded quality management framework for approximate computing, in which we enhance the energy efficiency by paying more proper computation effort of quality detection on different inputs. To achieve this, a light-weight structure supporting progressive detection and a heuristic algorithm to explore most energy-efficient topology are proposed. The experiment results show that our solution can achieve average 25.4% higher energy benefits over state-of-art techniques.

## Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61532017, 61572470, 61432017, 61521092, 61376043, and in part by Youth Innovation Promotion Association, CAS under grant No.Y404441000. The corresponding authors are Guihai Yan and Xiaowei Li.

## References

[1] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.
[2] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Computing approximately, and efficiently," DATE '15, 2015.
[3] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard, "Using code perforation to improve performance, reduce energy consumption, and respond to failures," 2009.
[4] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *MICRO*, pp. 449–460, IEEE Computer Society, 2012.
[5] X. He, G. Yan, F. Sun, Y. Han, and X. Li, "Approxeye: Enabling approximate computation reuse for microrobotic computer vision," 2017.
[6] X. He, W. Lu, G. Yan, Y. Han, and X. Li, "Exploiting the potential of computation reuse through approximate computing," *IEEE Transactions on Multi-Scale Computing Systems*, 2016.
[7] Y. Tang, Y. Wang, H. Li, and X. Li, "Approxpim: Exploiting realistic 3d-stacked dram for energy-efficient processing in-memory," IEEE, 2017.
[8] M. Imani, A. Rahimi, and T. S. Rosing, "Resistive configurable associative memory for approximate computing," in *DATE*, pp. 1327–1332, IEEE, 2016.
[9] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *HPCA*, 2017.
[10] X. He, W. Lu, G. Yan, and X. Zhang, "Joint design of training and hardware towards efficient and accuracy-scalable neural network inference," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2018.
[11] X. He, W. Lu, G. Yan, and X. Zhang "AxTrain: Hardware-Oriented Neural Network Training for Approximate Inference," in *ISLPED*, ACM, 2018.
[12] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: An online quality management system for approximate computing," in *ISCA*, pp. 554–566, IEEE, 2015.
[13] T. Wang, Q. Zhang, N. S. Kim, and Q. Xu, "On effective and efficient quality management for approximate computing," in *ISLPED*, pp. 156–161, ACM, 2016.
[14] D. Mahajan, A. Yazdanbakhsh, J. Park, B. Thwaites, and H. Esmaeilzadeh, "Towards statistical guarantees in controlling quality tradeoffs for approximate acceleration," in *ISCA*, pp. 66–77, IEEE Press, 2016.
[15] C. Xu, X. Wu, W. Yin, Q. Xu, N. Jing, X. Liang, and L. Jiang, "On quality trade-off control for approximate computing using iterative training," in DAC '17, (New York, NY, USA), pp. 52:1–52:6, ACM, 2017.
[16] Y. Wang, H. Li, and X. Li, "Real-time meets approximate computing: An elastic cnn inference accelerator with adaptive trade-off between qos and qor," DAC '17, 2017.
[17] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *ICCAD*, pp. 48–54, IEEE Press, 2013.
[18] W. Baek and T. M. Chilimbi, "Green: a framework for supporting energy-conscious programming using controlled approximation," in *ACM Sigplan Notices*, vol. 45, pp. 198–209, ACM, 2010.
[19] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "Sage: Self-tuning approximation for graphics engines," in *MICRO*, pp. 13–24, ACM, 2013.
[20] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in DAC '15, (New York, NY, USA), pp. 67:1–67:6, ACM, 2015.
[21] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pp. 475–480, IEEE, 2016.
[22] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.
[23] N. Binkert, B. Beckmann, G. Black, and Reinhardt, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.
[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, pp. 469–480, IEEE, 2009.