PAxC: A Probabilistic-oriented Approximate Computing Methodology for ANNs

Pengfei Huang, Chenghua Wang, Ke Chen and Weiqiang Liu

College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China
E-mail: {pfhuang, chwang, chen.ke and liuweiqiang}@nuaa.edu.cn

***Abstract*—In spite of the rapidly increasing number of approximate designs in circuit logic stack for Artificial Neural Networks (ANNs) learning. A principled and systematic approximate hardware incorporating domain knowledge is still lacking. As the layer of ANN becomes deeper, the errors introduced by approximate hardware will be accumulated quickly, which can result in unexpected results. In this paper, we propose a probabilistic-oriented approximate computing (PAxC) methodology based on the notion of approximate probability to overcome the conceptual and computational difficulties inherent to probabilistic ANN learning. The PAxC makes use of minimum likelihood error in both circuit and application level to maintain the aggressive approximate datapaths to boost the benefits from the trade-off between accuracy and energy. Compared with a baseline design, the proposed method significantly reduces the power-delay product (PDP) with a negligible accuracy loss. Simulation and a case study of image processing validate the effectiveness of the proposed methodology.**

***Index Terms*—Approximate computing, probabilistic-oriented, ANN learning, hybrid approximate circuits**

I. Introduction

As emerging computation-intensive applications, the ever-increasing performance requirements will soon exceed the growth of resource budgets. One of the most promising paradigms is the approximate computing (AxC) which has gained a increasingly significant attention. By considering such error tolerance, AxC pursues the imperfect designs to trade off accuracy for better energy efficiency. Specifically, researchers have developed a number of approximate circuit units (such as approximate multipliers [1] and adders [2]). Many recent work using approximate schemes at the hardware level shows that it can play a vital role in improving the overall system performance [3].

In fact, there is no universal approximate design that can fit all error-resilient applications. A systematic methodology is proposed to explore joint design space and find energy–area optimal solutions together with hardware parameters in [4]. A larger scope of applications can benefit from neural approximate computing which involves a great number of multiplications [5]. The fundamental computation unit in NN is a neuron, which performs a multiply-accumulate (MAC) operation. The most power consuming operation among NN computation is multiplications [1]. In this paper, we propose a PAxC methodology that allows the exploration of approximate multipliers (AMs) efficiently coordinating with ANN learning. By injecting the probabilistic energy benefit equation into the loss function to reflect the impact of noise introduced by approximated hardware on quality estimation, a more aggressive and adaptive approximation strategy can be found in a great number of optional datapaths. The main contributions of this paper are summarized as follows:

- We propose a probabilistic-oriented approximate computing methodology for ANNs.
- In ANN inference, we propose two maximum likelihood approximate computing algorithms using weight stationary in ANNs.
- By considering the accumulation and counterbalance of error along from the circuit to application, approximation friendly multiply-accumulate (MAC) designs are proposed for building the processing elements (PEs) in ANN learning.

II. Related Work

This section first introduces the related works on error analysis of both ANN learning and approximate hardware design for this work.

A. ANN Learning

ANNs are based on a set of connected units or nodes called artificial neurons, which loosely model the neurons in the biological brain. A multilayer perceptron (MLP) is a typical type of feedforward ANNs. MLPs are very useful in research because they can solve problems stochastically, which usually allows approximate solutions to extremely complex problems, such as fitness approximation.

B. Comprehensive Metrics for Approximate Hardware Design

More acceleration leads to more accuracy deterioration. In order to obtain the optimal trade-off, error analysis and effective metrics should be considered in advance. To make a trade-off between preciseness and energy efficiency, illustrative figure of merit (FOM) and FOM1 are suggested in [6] and [7] respectively.

III. The Proposed PAxC Methodology

Multiple approximate units usually interact in a datapath; moreover, many applications often require complex datapaths rather than just a single operation (such as multiplication), so

978-3-9819263-6-1/DATE22/©2022 EDAA

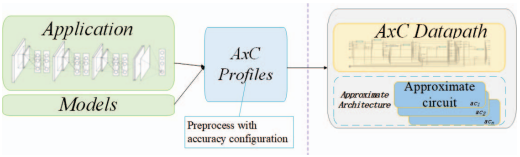

Fig. 1: Overview of the proposed PAxC.

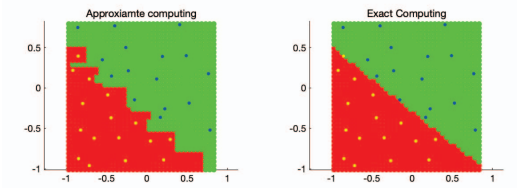

Fig. 2: The generalization result of the model from (a) approximate and (b) exact training.

the optimal hybrid scheme is quite essential to calibrate the accumulative error. Fig. 1 gives an overview of the proposed PAxC methodology. Although machine learning applications have error resilience capabilities, AxC should be applied in a principled manner to ensure that the impact on output quality is negligible (or acceptable).

A. PAxC for ANN Inference

Inference applies knowledge from a trained neural network model and uses it to infer a result. When a new unknown data set is input through a trained neural network, it outputs a prediction based on predictive accuracy of the neural network. Inference comes after training as it requires a trained neural network model. An application supports lots of approximate settings with multiple approximate modules. This makes computation-intensive retraining impossible to pick up a "best" strategy among all the approximate strategies. Meanwhile, excessive retraining leads to over-fitting of the noises. In Fig. 2 (a), blue and yellow dots (training data) are perfectly classified. It indicates that a high accuracy for training data is not necessarily a good indicator, as it may also imply that the model is suffering from overfitting. Such a data set may perform well in the test scenario, but it may fail in specific applications since it lacks the generalization ability as shown in Fig. 2 (b). Hence, it is reasonable to find the optimal combinations for approximate multipliers to inference on the model trained by exact computation.

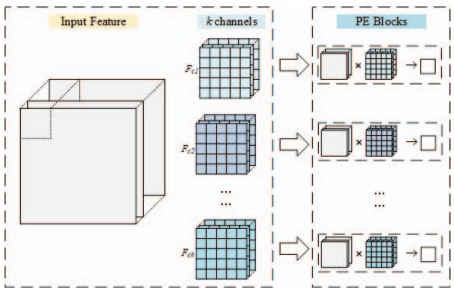

Fig. 3: Approximation friendly PE blocks.

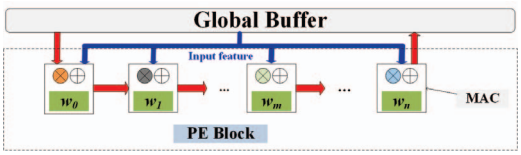

Fig. 4: Approximation friendly MACs.

1) Approximation Friendly Multiply-accumulate (MAC) Designs: In many ANNs, the computation of score is actually the dot product of the feature ($\vec{x}$) and the weight ($\vec{w}$) (i.e., $\sum_i w_i x_i$). In the inference process, the MAC operations of the feature extraction (convolutional layer) and the classification can be easily parallelized. To increase the design flexibility and facilitate approximation, we propose a block by block neural network architecture design, which includes the following two basic computing blocks.

A) Processing Element (PE) Block: In convolution process, the core operation is multiplying the sliding filter and the corresponding sub-block of the input feature. These operations can be conveniently parallelized in different channels and filters. Hence, operations in each filter can be constructed as one PE, and operations of channels can be divided into separated PE blocks. As shown in Fig. 3, the PE in one block maintains the same filters which can simplify the complexity of PE design.

B) Multiply-Accumulate (MAC) Block: The elementary unit of ANN is MAC which only contains an addition and multiplication. Each individual MAC can use a unique approximate strategy. As shown in Fig. 4, MACs in one PE use different approximate multipliers. The output of the PE is the accumulation of n MACs, hence, the optimal group of the approximate multipliers should compensate each other to achieve the overall fine result.

2) PAxC Algorithms: Since one operand of the multiplication is the weight w_i which is predefined by the training model. Intuitively, it is easy to estimate the error rate $P_e(x|w_i)$ introduced by approximate multiplier based on the conditional probability of w_i with random input x. In order to search the optimal approximate multiplier $\mathcal{M}_k$ for the static w_i, performance $P(\mathcal{M}|x,w)$ and related FOM ($RFOM$) are defined as follows:

$$P(\mathcal{M}_k|w_i,x) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{RFOM(\mathcal{M}_k|w_i,x_j)} \quad (1)$$

$$RFOM = \frac{\alpha \times PDP \times area}{\beta \times (1-NRED)} \quad (2)$$

$$\hat{\mathcal{M}}_k = \arg\max_{\mathcal{M}_k \in \Theta} \hat{P}(\mathcal{M}_k|w_i,x) \quad (3)$$

where, $NRED$ is normalized related error distance, α and β are the coefficients to balance the energy saving and accuracy loss. $RFOM(\mathcal{M}_k|w_i,x_j)$ is the $RFOM$ under the given approximate multiplier $\mathcal{M}_k$, weight w_i and input x_j. The goal of

maximum likelihood estimation is to find the optimal $\hat{\mathcal{M}}_k$ that maximize the likelihood performance function over the potential approximate multiplier space Θ. The overall probabilistic-oriented approximate computing algorithm (PAxC1) is summarized in Algorithm 1 which finds out the maximum likelihood of approximate operators. To better facilitate approximation and reduce the data movement between the AI engine and the memory, the parameters of the involved filter will be reused in the PE block with proposed approximation friendly MACs. In order to neutralize the error on final summation of MACs for the output feature, a neighborhood sensitive performance $\hat{P}$ is defined as follows:

$$\hat{P}(\mathcal{M}_k, \mathcal{M}'_k|w_i, x) = \frac{1}{RFOM(\mathcal{M}_k|w_i, w_{i-1}, x)} + \frac{1}{RFOM(\mathcal{M}'_k|w_i, w_{i+1}, x)} \tag{4}$$

The effective pair-wise accelerators can be determined via the optimal combination performance $\hat{P}$. Similarly, the maximum likelihood $\hat{w}_i$ of weight w_i can be obtained under given approximate operator $\hat{\mathcal{M}}^i$. Suppose $\mathcal{C}$ is the correct operator.

$$|\hat{\mathcal{M}}(w, x) - \mathcal{C}(w, x)| = \varepsilon \tag{5}$$

$$\begin{aligned} \arg\min_{\hat{w}} |\hat{\mathcal{M}}(\hat{w}, x) - \mathcal{C}(w, x)| &= \hat{\varepsilon} \\ (\exists \hat{w}) \quad s.t. \quad \hat{\varepsilon} &\leq \varepsilon \end{aligned} \tag{6}$$

From Eqs. (5) and (6), weight w_i can be fine-tuned as $\hat{w}_i$ to formulate a more approximation friendly model of the neural network in a regression way. In the extreme case ($\hat{\varepsilon} == \varepsilon$), $\hat{w}_i$ is equal to w_i. Algorithm 2 presents the steps of the fine-tuned probabilistic-oriented approximate computing algorithm.

Algorithm 1 PAxC1 for ANNs

Require:
The set of weight W;
The set of Data $\mathcal{X}$;
The set of approximate operators $\mathcal{M}$;
The setting of the coefficients (α and β).
Ensure:
A maximum likelihood performance $\hat{\mathcal{M}}$ for each $w_i \in W$
1: Set $i := 0$;
2: **repeat**
3: Set $i \leftarrow i + 1, \hat{\mathcal{M}}^i \leftarrow \mathcal{M}_1$;
4: **repeat**
5: **if** $P(\mathcal{M}_k|w_i, x) > P(\mathcal{M}_{k-1}|w_i, x)$ and $k > 1$ **then**
6: Update $\hat{\mathcal{M}}^i := \mathcal{M}_k$;
7: **end if**
8: **until** Each potential approximate operator $\mathcal{M}_k \in \mathcal{M}$ is traversed.
9: **until** Probabilistic-oriented approximate operator $\hat{\mathcal{M}}$ for all weights are found.

IV. Evaluation and Analysis

Hardware metrics, such as power consumption, area, critical path delay and PDP are considered in this section.

Algorithm 2 Fine-tuned PAxC2 for ANNs

Require:
The set of weight W;
The set of Data $\mathcal{X}$;
The set of approximate operators $\mathcal{M}$;
The setting of the coefficients (α and β).
The set of maximum likelihood performance $\hat{\mathcal{M}}$ for each $w_i \in W$
Ensure:
The set of fine-tuned weight $\widetilde{W}$
1: Set $i := 0$;
2: **repeat**
3: Set $i \leftarrow i + 1$;
4: **repeat**
5: **if** $P(\hat{\mathcal{M}}|w_j, x) > P(\hat{\mathcal{M}}|w_i, x)$ **then**
6: Update $w_i := w_j$;
7: **end if**
8: **until** The other weights (w_j) are traversed.
9: **until** All weights are fine-tuned under the maximum likelihood performance.

A. Simulation Setup

Although the proposed method is applicable to most reconfigurable multipliers and adders, the method in [1] is considered as multipliers are more complex and have higher energy consumption than adders. Digits dataset in the standard UCI datasets are used. Logic synthesis and simulation tools i.e. the Synopsys Design Compiler and VCS are utilized; the target synthesis technology is given by the Nangate 45-nm open cell library.

B. Simulation Results

A typical MLP with 3 layers is built to classify the digits data-set. For convenience, both α and β are set as 1 in PAxC algorithms. In order to analysis the superiority of our proposed method, the pre-processing and post-processing of an application are omitted. The total number of weights is 7620 and the hardware consumption result is the average values of overall approximate multipliers embedded in the network. Table I summarizes the accuracy loss, power, delay, area and power-delay product. Both PAxC1 and PAxC2 maintain better energy performance than the fixed AMs (p=2,6), and certainly more accuracy than fixed AM (p=10). After weights fine-tuned, PAxC2 achieves higher accuracy than PAxC1 with the exact same hardware design.

C. Case Study: Image Processing

An autoencoder is a type of ANNs used to learn efficient codings of unlabeled data. The autoencoder learns a representation for a set of data, by training the network to ignore insignificant data ("noise"). Exact, PAxC and the moderate accelerator (p=6) in [1] are applied to auto-encode the images in Fig. 5. The models is trained by exact computing. Since the autoender involves many multiplications, the accumulated error has a large impact. From the results, all decoding images of the

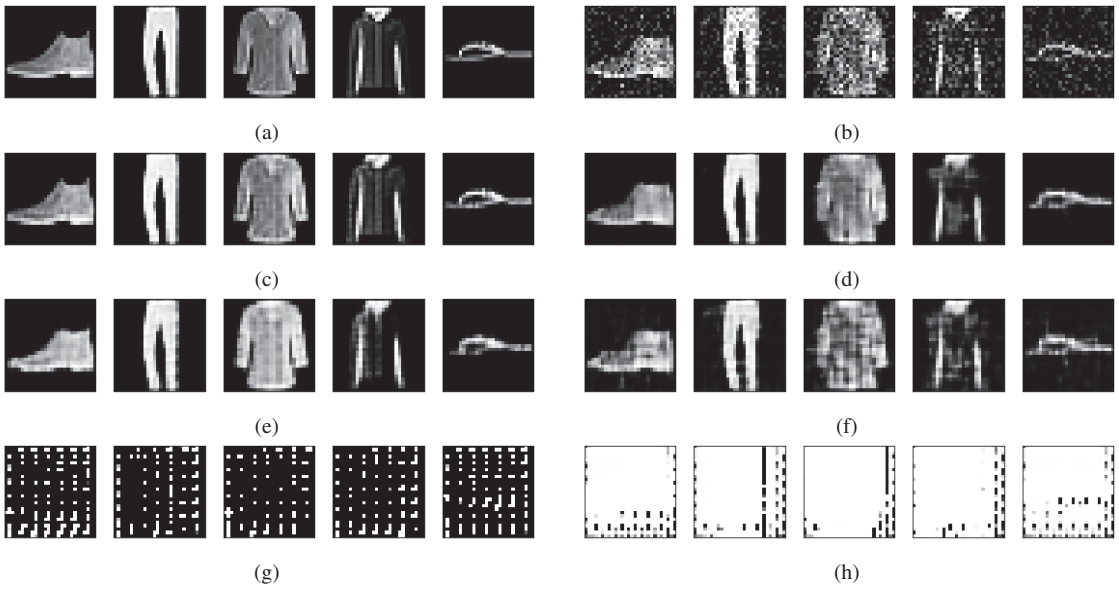

Fig. 5: Image processing: (a) original images, (b) original images with noise, (c), (e), (g) are the reconstructions of original images by auto-encoders with exact computing, PAxC and AM(p=6) [1]; (d), (f), (h) are the reconstructions of noisy images with exact computing, PAxC and AM(p=6) [1] receptively.

TABLE I: Accuracy and Energy Evaluation

Approximate Multipliers	Accuracy loss (%)	Power (mW)	Delay (ps)	Area (mm^2)	Energy (nJ)
PAxC1	0.35	33.396	0.63	0.1327	21.039
PAxC2	0.13	33.396	0.63	0.1241	21.039
AM (p=2) [1]	0.28	44.385	0.63	0.1592	27.963
AM (p=6) [1]	2.33	36.855	0.61	0.1373	22.482
AM (p=10) [1]	10.20	25.319	0.58	0.1009	14.685
Exact multiplier	0	48.179	0.71	0.2011	34.207

accelerator (p=6) in [1] suffer too much deviation caused by the approximation. our proposed method can achieve almost exact in a dedicated approximate strategy with maximum likelihood performance and compensation group. Input noises can be denoised through the learning filters which are not aware of the noise introduced by the hardware approximators. Our method can provide the accelerators which are adapted to weights of filters with injected errors. Hence, it can achieve the desired performance with acceptable accuracy loss.

V. Conclusion

By allowing the almost exact computation in error-tolerance applications, approximate computing can gain both performance and energy efficiency. In this work, we propose a probabilistic-oriented approximate computing methodology and an approximate friendly hardware architecture for ANN. Based on the notion of approximate probability, we designed two algorithms to search the optimal combinations of approximate operators and fine-tuned weights. Approximation friendly MAC and PE block are designed to better apply the hybrid accelerators. The proposed PAxCs aggressively employ approximate accelerators under the compressive metric RFOM, thus achieving better performance, while ensuring that the result quality meets the application's requirements. Our evaluation showed that the proposed PAxCs reduce power consumption and maintain high quality.

Acknowledgment

The authors would like to acknowledge support from National Natural Science Foundation of China under grant No. 62022041 and 61871216.

References

[1] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 Booth multipliers for error-tolerant computing," *IEEE Trans. Computers*, vol. 66, no. 8, pp. 1435–1441, 2017.

[2] M. A. Hanif, R. Hafiz, O. Hasan, and M. Shafique, "Pemacx: A probabilistic error analysis methodology for adders with cascaded approximate units," in *Proc. DAC*, 2020, pp. 1–6.

[3] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Block-based carry speculative approximate adder for energy-efficient applications," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 137–141, 2020.

[4] T. Alan, A. Gerstlauer, and J. Henkel, "Cross-layer approximate hardware synthesis for runtime configurable accuracy," *IEEE Trans. VLSI Systems*, vol. 29, no. 6, pp. 1231–1243, 2021.

[5] H. R. Mahdiani, M. Haji Seyed Javadi, and S. M. Fakhraie, "Efficient utilization of imprecise computational blocks for hardware implementation of imprecision tolerant applications," *Microelectronics Journal*, vol. 61, pp. 57–66, 2017.

[6] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. VLSI Systems*, vol. 25, no. 4, pp. 1352–1361, 2017.

[7] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4200–4208, 2019.