COIN: Combinational Intelligent Networks

Igor D. S. Miranda[1], Aman Arora[2], Zachary Susskind[2], Josias S. A. Souza[1],
Mugdha P. Jadhao[2], Luis A. Q. Villon[3], Diego L. C. Dutra[3], Priscila M. V. Lima[3],
Felipe M. G. França[4], Mauricio Breternitz Jr.[5], and Lizy K. John.[2]

[1]Universidade Federal do Recôncavo da Bahia, Brazil. [2]University of Texas at Austin, USA.
[3]Universidade Federal do Rio de Janeiro, Brazil. [4]Instituto de Telecomunicações, Portugal.
[5]ISCTE Instituto Universitário de Lisboa, ISTAR, Portugal.

***Abstract*—We introduce Combinational Intelligent Networks (COIN), a machine learning technique that targets edge inference using low-resourced FPGAs or ASICs. COIN is an improvement on LogicWiSARD, a recent weightless neural network that achieves low power, small area, and high throughput. We convert the LogicWiSARD model into a binary neural network, train it using backpropagation, and then convert it to a COIN model. As a result, COIN can achieve higher accuracy than LogicWiSARD or it can require significantly fewer hardware resources when comparing models with similar accuracies. In comparison to a BNN implementation, FINN, small and large COIN models are more energy efficient demonstrating up to 11.5x higher inferences/Joule at similar accuracy. Our tool executes the complete flow, from training to RTL, and is publicly available.**

***Index Terms*—Weightless neural networks, LogicWiSARD, binary neural networks, FPGA, ASIC**

I. Introduction

Deploying AI on edge devices poses unique challenges not seen in other domains. Edge devices have aggressive area and energy constraints, yet may be expected to maintain low latencies and high throughputs. Traditional deep neural networks may be infeasibly expensive in this domain, even when techniques such as pruning and quantization are used.

An attractive alternative in edge contexts are binary neural networks (BNNs), which use single-bit weights and activations representing values -1 and 1 [1]. Weightless neural networks (WNNs) go a step further by almost eliminating arithmetic operations with reduced memory requirements [2], [3].

This work[†] introduces a new approach to the training of WNNs using the BNN framework. It is done by converting a LogicWiSARD model to an equivalent BNN model, training it with backpropagation, and then converting it to a new WNN model. Our contributions in this work are the following:

1) A novel approach to WNN models that uses backpropagation to enhance model accuracy and can be implemented using only logic functions (no arithmetic, no memory). This leads to efficient hardware implementations. We call this architecture **CO**mbinational **I**ntelligent **N**etworks (COIN);
2) A new encoding technique to be used with WNNs that reduces resources utilization;
3) Evaluation of the COIN implementation with existing state-of-the-art WNN and iso-accurate BNN models;
4) An open-source tool chain to automatically generate these models from training to RTL. This tool chain is available at: *https://github.com/lasdi/coin*

[†]This work was supported by the Fulbright program and FCT/MCTES projects UIDB/50008/2020, UIDP/4466/2020, and UIDB/04466/2020.

II. Proposal: COIN

A. LogicWiSARD training

For LogicWiSARD training, we used the same methodology introduced in [2].

A trained LogicWiSARD model is defined by its minterms m_n^r and truth table outputs $\tau_{n,k}^r$, where r is the RAM index, $n = 0,...,N^r-1$ is the minterm index in a RAM with N^r entries, and k is the class index. After encoding and mapping, an input x produces R addresses x^r, which are sent to classification. For a K-class LogicWiSARD model, the predicted class can be evaluated as $\widehat{C} = argmax(\theta_k)$ where

$$\theta_k = \sum_{r=0}^{R-1}\sum_{n=0}^{N^r-1} \tau_{n,k}^r q(m_n^r, x^r), \tag{1}$$

and

$$q(a,b) = AND_{unary}(XNOR_{bitwise}(a,b))). \tag{2}$$

θ_k corresponds to the score of the k_{th} discriminator. $q(a,b)$ function is used to return 1 if its arguments are coincident and 0 otherwise. In our experiments, we could observe that pruning LogicWiSARD model lowers its accuracy but does not affect final COIN accuracy significantly.

B. LogicWiSARD to BNN conversion

A BNN without hidden layers, similar to a multinomial logit, can represent a LogicWiSARD model. The number of inputs of this BNN model is the total number of minterms, which can be defined as

$$N = R\sum_{r=0}^{R-1} N^r. \tag{3}$$

The inputs for the BNN model should be derived from the original LogicWiSARD inputs as follows

$$x_l' = q(m_n^r, x^r), \tag{4}$$

where

$$l = n + \sum_{r'=0}^{r-1} N^{r'}. \tag{5}$$

DOI 10.1109/ASAP57973.2023.00016

l is a unified index to refer to all minterms across RAM nodes which are used as inputs for the equivalent BNN. Then, the following BNN classifier can be used for training

$$y'_k = BatchNorm(\theta'_k) \tag{6}$$

where

$$\theta'_k = \sum_{l=0}^{N-1} \omega_{k,l} x'_l, \tag{7}$$

BatchNorm is a batch normalization layer, and $\omega_{k,l}$ are trainable weights. Note that linear activation functions are used for the output neurons.

C. BNN to COIN conversion

The COIN model consists of implementing the BNN described in Eq. (7) using the same arrangement of Eq. (1), therefore

$$\theta'_k = \sum_{r=0}^{R-1} \sum_{n=0}^{N^r-1} \omega_{k,l} q(m_n^r, x^r), \tag{8}$$

A key difference in the COIN model is that $\omega_{k,l}$ values are $\{-1,+1\}$ instead of $\{0,1\}$ as in $\tau_{n,k}^r$. Additionally, batch normalization should be performed to (7) before the highest score is selected. Our batch normalization is performed as

$$\sigma_k^+ = \frac{\sigma_k}{\sqrt{\frac{1.5}{N+K}}} \tag{9}$$

where σ_k is the batch normalization method proposed by FINN [4] which is suited for binary outputs while requiring simpler hardware. We also include an extra normalization to compensate the Glorot initialization during BNN training.

Finally, a COIN classifier can be implemented as $\widehat{C} = argmax(\widehat{\theta}'_k)$ where

$$\widehat{\theta}'_k = \sum_{r=0}^{R-1} \sum_{n=0}^{N^r-1} (\omega'_{k,l} q(m_n^r, x^r) - \sigma_k^+). \tag{10}$$

D. Reverse ripple thermometer encoding

We introduce the reverse ripple thermometer (RRT) encoding, a method for use when input samples are in a positive range with approximately exponential distribution (or more values close to zero), with thresholds defined by power of two exponentials within the input range. The RRT encoding can be performed using a binary logarithm function where the number of bits set to 1 is $b = \lfloor Log_2(a) \rfloor$ for a given input a.

III. Experiments and Evaluation

A. Implementation Results and Comparison

We implement two COIN models, a small one ($n = 16$ and $T = 8$) and a large one ($n = 16$ and $T = 8$), referred to as COIN/small and COIN/large. Using the MNIST dataset [5], we compare COIN/small (95.8% acc.) with BNN/FINN-SFC (95.8% acc.) [4] and previous WNN work, namely BTHOWeN (95.2% acc.) [3] and LogicWiSARD (95.0% acc.) [2]. COIN/large (97.8% acc.) are compared with BNN/FINN-LFC (98.4% acc.) [4]. For COIN, we use RRT encoding which provided smaller hardware in our experiments. All these models were synthesized and implemented for the Xilinx FPGA device XC7Z045-FFG900 operating at 200MHz.

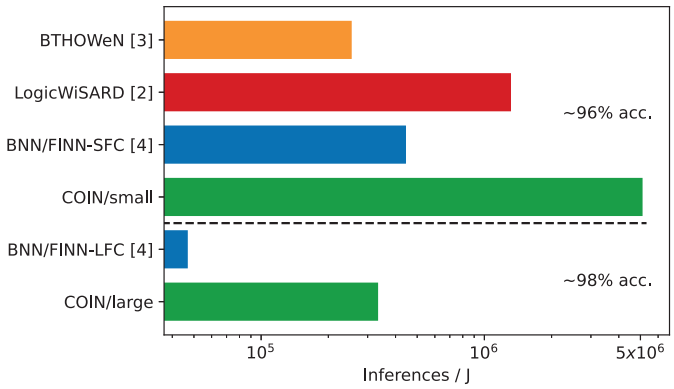

Fig. 1: Energy efficiency comparison between proposed method (COIN), BNN/FINN and previous WNN work in FPGA. LogicWiSARD, BTHOWeN, BNN/FINN-SFC and COIN/small models have accuracy in the 96% range while BNN/FINN-LFC and COIN/large are in the 98% range.

COIN/small requires 6805 LUTs and 184 FFs, which is significantly fewer resources than the other models in the same accuracy range. For example, COIN/small needs 91.4% fewer LUTs than LogicWiSARD. COIN/large also reduces hardware requiring 70470 LUTs and 226 FFs while BNN/FINN-LFC requires 82988 LUTs and 396 36-Kbit BRAMs. Due to the reduced hardware, COIN designs for MNIST have shown large improvements regarding energy efficiency, as shown in Fig. 1.

IV. Conclusion and Future Work

In this work, we introduce COIN, a machine learning technique that can be efficiently implemented in hardware. The proposed method enhances the state-of-the-art WNN approach, LogicWiSARD, with backpropagation training based on the BNN framework, improving the accuracy by 2.6%. Across all comparisons, the resource utilization is significantly less than in the previous work and the increase in the energy efficiency ranges between 3.9x and 11.5x. With improvements in accuracy, resource utilization, and energy efficiency, the proposed method is an alternative for edge inference using FPGAs or ASICs. However, studies should be conducted to further improve COIN accuracy for complex tasks.

References

[1] M. Courbariaux *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[2] I. D. S. Miranda *et al.*, "Logicwisard: Memoryless synthesis of weightless neural networks," in *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2022.

[3] Z. Susskind *et al.*, "Weightless neural networks for efficient edge inference," in *31st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2022.

[4] Y. Umuroglu *et al.*, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of FPGA'17*.

[5] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/