Energy Adaptive Convolution Neural Network Using Dynamic Partial Reconfiguration

Eman Youssef
Microelectronics Department
Electronics Research Institute (ERI)
Cairo, Egypt
emanyousef2014@eri.sci.eg

Hamed A. Elsemary
Computer Engineering Dept
Prince Sattam Bin Abdulaziz University
KSA
elsemary@gmail.com

Magdy A. El-Moursy
Design Creation Division
Mentor Graphics
Cairo, Egypt
magdy_el_moursy@mentor.com

Ahmed Khattab
Electronics and Communications Engineering
Faculty of Engineering, Cairo University
Giza, Egypt
akhattab@ieee.org

Hassan Mostafa
Electronics and Communications Engineering
Faculty of Engineering, Cairo University
Giza, Egypt
hmostafa@uwaterloo.ca

***Abstract*—Convolutional Neural Network (CNN) is a good candidate for computer vision applications. CNN is well known for its great classification accuracy at image recognition tasks. The cost of CNN is its large power consumption as it needs a lot of multiplication and addition operations. Approximation can reduce the power consumption. CNNs can be implemented by CPUs, GPUs or FPGAs. In this paper, the proposed CNN is implemented on Xilinx XC7Z020 FPGA and is trained to recognize MNIST dataset. This CNN is approximated through quantization which reduces the accuracy only by 0.53% while using 7-bits for the implementation. A reduction of 2.7X is achieved in energy consumption compared to the conventional design which uses 16-bits. Dynamic Partial Reconfiguration (DPR) reconfigures the FPGA during the run time with appropriate power consumption design if the battery level decreases. This enables recognition applications to run with low power budget but with sacrificing minor accuracy instead of termination.**

***Index Terms*—Convolutional Neural Network, Approximate Computing, Precision Scaling, MNIST, DPR.**

I. Introduction and Related Work

Deep learning networks are used to recognize speech, images and documents with high accuracy. Convolutional Neural Network (CNN) is a widely used deep learning network. CNN is used in image classification for providing the probability of the classes that describe the input image. CNN consists of a series of convolution (Conv) layers followed by fully connected (FC) layers. Each convolution layer is followed by a pooling layer. A fully connected layer connects all the neurons in the previous layer to each neuron in the current layer.

Previous CNN works were applying recognition in different applications and datasets. In [1], LeNet-5 is proposed to recognize hand written digits. In [2], AlexNet is proposed to recognize Image-Net dataset. Neural networks need a huge number of multiplication operations to achieve its recognition task which consumes a lot of power. As a result of the large power consumption, many research works targeted developing approximate CNN to reduce the consumed power. In [3], neural networks are trained to make the network weights and activations in a binary form. There are previous works on simplifying the neural networks complexity through pruning [4]. In [5] and [6] approximate multipliers are proposed. In [7], a prediction strategy is used at the convolution layer to reduce the power consumption. In [8], the number of needed operations to recognize the input image is decreased. In [9] and [10], the non-critical neurons of an Artificial Neural Network (ANN) are approximated. The authors of [11] proposed singular value decomposition approximation (SVDA) method to transfer 2D convolution to pairs of 1D convolutions which reduces the number of multiplications. In [12], ANN is implemented on FPGA using on chip memory only. In [13], new scalable FPGA design is implemented on FPGA using HLS.

In this work, the proposed CNN is trained to recognize MNIST dataset. Then, the resulting parameters are approximated using precision scaling while using a different bitwidth to represent each parameter. This reduces the power consumption for the whole CNN hardware. The approximated CNN is implemented on FPGA while using different bitwidth, then Dynamic Partial Reconfiguration (DPR) is used to reconfigure FPGA during the run time.

This paper is organized as follows. Section II presents the proposed CNN architecture. Section III explains Dynamic Partial Reconfiguration. Section IV discusses the experimental results. Finally, the conclusion is propounded in Section V.

II. Proposed Approximated CNN Architecture

The proposed EEPS-CNN-1 architecture for MNIST dataset recognition has two convolution layers and two fully connected layers. All convolution layer outputs are passed to a ReLU (Rectified Linear Unit) activation function. Each convolution layer is followed by MaxPooling layer with dimension 2×2 and a stride length of two. The number of filters is 2 filters and 4 filters for the first and second convolutional layers, respectively. The input image to the input convolution

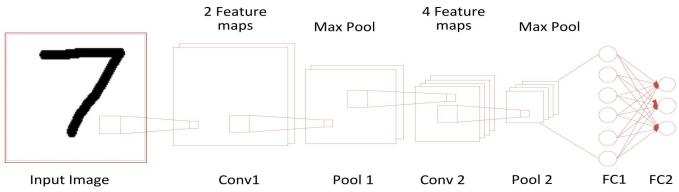

Fig. 1. EEPS-CNN-1 Architecture.

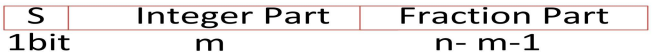

Fig. 2. Available n bitwidth to represent each network parameter.

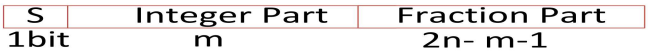

Fig. 3. Available $2n$ bitwidth to represent each network parameter.

layer is padded to preserve its spatial size. The dimension of all filters is 3×3 with a stride length of one. A very small filter size captures the fine details of the image and the image dimension is small, hence, a 3×3 filter size is used. The number of neurons is 20 and 10 for the first and the second fully connected layers, respectively. The first fully connected layer is followed by *ReLU* activation function given by [14]

$$f(z) = max(0, z) \tag{1}$$

The second fully connected layer is followed by *softmax* activation function to convert its output to probability distribution [15]. A fewer number of filters with less kernel dimensions (3×3) at the convolution layers and less number of neurons are used at the fully connected layers to reduce the number needed of multiplication operations. Our CNN architecture is shown in Figure 1. The parameters for EEPS-CNN-1 are shown in Table I where *maps* refers the number of extracted features at the convolution layer. The used number of layers, filters and neurons are selected based on a trade-off between network accuracy and the number of multiplications.

TABLE I
EEPS-CNN-1 PARAMETERS.

Layer Type	Feature Maps SIZE	Number of parameters
Input Image	28x28	-
Convolution 1	28x28 2 maps	20
Pooling 1	14x14 2 maps	-
Convolution 2	12x12 4 maps	40
Pooling 2	6x6 4 maps	-
Fully Connected 1	20	2900
Fully Connected 2-Output	10	210

A. EEPS-CNN-1 Approximation Via Precision Scaling

Approximate computing through quantization achieves power reduction for digital circuits [16]. Quantization reduces the switching activity by computing using fewer bits. Reducing the switching activity reduces the dynamic power consumption. For adders, registers and comparators, the dynamic power is linearly reduced. While for multipliers, the dynamic power is quadratically reduced [17]. The network parameters are quantized and represented with n-bits using a fixed point number representation. The same number of bits is used for all CNN layers which is called uniform quantization as introduced in [17]. The parameters of each layer should be examined separately to find the maximum and the minimum values for each layer to divide the available bitwidth n as shown in Figure 2, where m is the number of bits needed to represent the integer part.

After obtaining the trained weights from the training step and quantizing these weights, the proposed CNN architecture is tested using these quantized weights. For the test function, repetitive truncation is applied after each addition step to guarantee that the accumulated results can be represented with $2n$ bits with the fraction part represented using $2n - m - 1$ bits as shown in Figure 3. Next, the final accumulated result which is represented by $2n$ bits is truncated to n-bits to use the same hardware with the same bitwidth for all layers.

B. Hardware Architecture

The proposed hardware architecture to recognize MNIST dataset is shown in Figure 4. The architecture has memory units to save the image under test, CNN weights and intermediate results from each layer. Memory Access units are used to prepare the needed nine operands to perform convolution operations from the image memory and the intermediate results memory. A computation unit is used to perform multiplication and addition operations at the convolution and fully connected layers. ADD2 Units are used to accumulate the results of the registers. MaxPool Units are used to perform MaxPool layer operation which has a comparison between four parameters. ReLU Units are used to perform the function of ReLU activation function. The register file is used to restore the layer outputs before restoring to the memory. A comparator is used to compare between the final ten outputs resulting from the last fully connected layer. All operations are done using n-bits fixed point arithmetic units instead of conventional 32-bit floating point units, where n equals to 16, 12, 10, 8, 7, 6 and 5 bits to reduce the dynamic power dissipation.

III. DYNAMIC PARTIAL RECONFIGURATION

DPR is a new feature offered by modern FPGAs to solve the limited hardware resources problem [18]. DPR allows the reconfiguration of the FPGA programmable logic during the run time. DPR hardware design is divided into a static part and a dynamic part. The static part is configured at booting time using a full bit-stream. The dynamic part contains one or more Reconfigurable Partition (RP). At the run time, each RP can be reconfigured with different partial bit-streams without changing the static part. The needed hardware resources are reduced through sharing the same programmable logic between multiple Reconfigurable Modules (RM).

The FPGA reconfiguration time is an important factor in DPR. The reconfiguration time is proportional to the size of

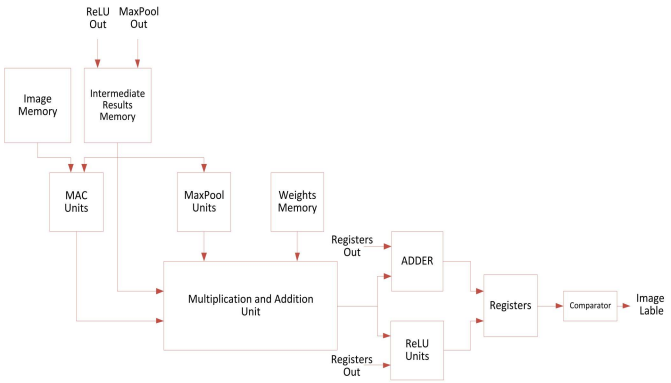

Fig. 4. EEPS-CNN-1 hardware implementation.

the partial bit-stream. As the size of the partial bit-stream increases, the reconfiguration time increases. The size of the partial bit-stream depends on the size of the region to be reconfigured.

In this paper, DPR is used to reconfigure the FPGA according to the appropriate power level design. The block diagram of the proposed DPR system is shown in Figure 5. The processing system is used to transfer the required partial bit-streams from DDR to the Internal Configuration Access Port (ICAP). ICAP is used to reconfigure RP. The required partial bit-streams are determined according to the available battery power.

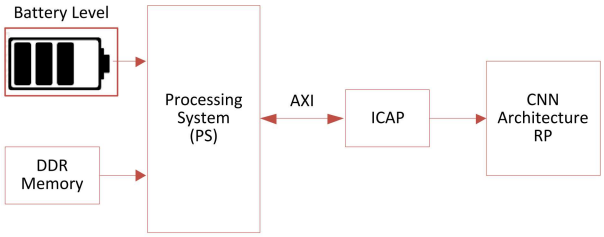

Fig. 5. DPR system block diagram.

IV. Experimental Results and Discussions

A. EEPS-CNN-1 Training

Python programming language is used to train the proposed CNN architecture to recognize MNIST dataset [19]. There are 60,000 samples for training and 10,000 samples for testing. The 60,000 training samples are divided into 52200 training set and 7800 validation set. All images are normalized to limit the range of data between 0 and 1. Adadelta optimization algorithm is used in training with 32 batch size [20]. The test accuracy for MNIST is 98.12%. During training, all network parameters are represented by 32-bit floating point numbers.

B. EEPS-CNN-1 Precision Scaling Results

The efficiency of the proposed approximation is evaluated using MNIST dataset using Python programming language. The network parameters and the inputs are quantized and represented with n-bit fixed point number representation where n is equal to 16, 12, 10, 8, 7, 6 and 5 as explained earlier. The output of each layer is examined. For MNIST dataset, m is equal to 4 ,5 ,6 and 8 for Conv1 output, Conv2 output, FC1 output and FC2 output, respectively. The resulting accuracy and accuracy loss are shown in Table II. The accuracy loss is the difference between of the accuracy of using 16-bits fixed point and the accuracy of using n-bits. The accuracy normalized to the fully accurate CNN is shown in Figure 6.

TABLE II
CNN ACCURACY AND ACCURACY LOSS FOR MNIST DATASET

Number of Bits	Accuracy	Accuracy Loss
16-bits	98.12 %	0 %
12-bits	98.12 %	0 %
10-bits	98.08 %	0.04%
8-bits	97.87 %	0.25 %
7-bits	97.60 %	0.53 %
6-bits	96.07 %	2.09 %
5-bits	81.71 %	16.72 %

C. Hardware Architecture Results

The performance of the approximated convolutional neural network architecture is investigated in this section. VHDL language is used to model the proposed hardware architecture on Xilinx Vivado (v.2015.2). The proposed hardware architecture is implemented on xc7z020clg484-1 FPGA at a Zynq-7000 evaluation board recognize MNIST. In Table III, the FPGA resource utilization is shown while using different bitwidths. At 50 MHz system clock frequency, the proposed design can recognize 3645 image per second. The needed time to recognize one image is 13715 cycles $\times$ 20 ns = 0.27 ms. The power consumption and energy while decreasing the number of bits are shown in Table IV. The hardware architecture achieves reduction in energy equals to 1.46X, 1.86X, 2.41X, 2.73X, 3.42X and 3.73X for 12, 10, 8, 7, 6 and 5 bits with respect to the original 16-bits case. Figure 7 shows the energy reduction while decreasing the number of used bits. The FPGA resource utilization while using DPR is shown in Table V. The reconfiguration time is equal to 127 ms as the ICAP processor throughput is 10 *MBps* and the file size is equal to 1.27 *MB*.

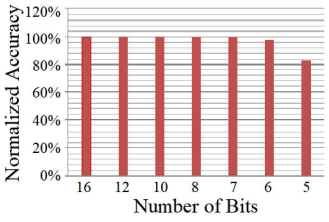

Fig. 6. Normalized accuracy of EEPS-CNN-1.

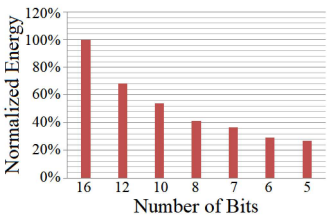

Fig. 7. Reduction in energy.

V. Conclusions

In this paper, quantization has been used to reduce the CNN energy consumption through reducing the number of bits to represent CNN parameters. Approximated CNN has been evaluated using MNIST dataset. The proposed architecture has

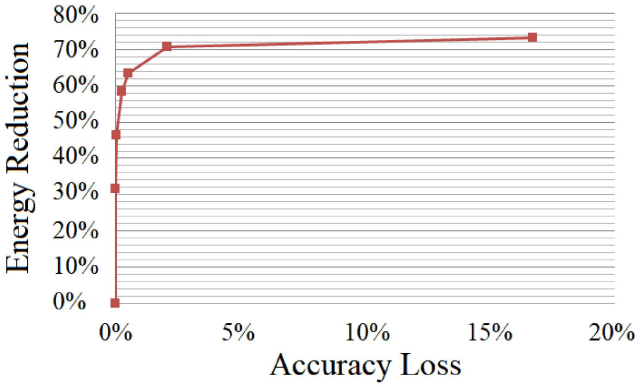

Fig. 8. Accuracy loss vs. energy reduction with approximated CNN.

TABLE III
FPGA RESOURCE UTILIZATION FOR MNIST WITH DIFFERENT BITWIDTH

Number of Bits	FF	LUT	Memory LUT
16	8466	17791	1792
12	6362	12060	1344
10	5310	9881	1120
8	4258	7434	896
7	3732	6011	784
6	3206	5296	672
5	2677	4491	560
Available	106400	53200	17400

been implemented using Xilinx Vivado (v.2015.2) and implemented on FPGA. The presented experiments has demonstrate 2.7x reduction in energy consumption with a maximum of 0.53% loss in CNN accuracy for MNIST while using 7-bits to represent all network parameters, as compared to 16-bits. DPR enables the images recognition process to continue using low power design at the expense of recognition accuracy instead of termination if the available power level decreases.

TABLE IV
POWER AND ENERGY CONSUMPTION PER IMAGE FOR CNN.

Number of Bits	Power (mw)				Energy/Image (μJ)
	Clocks	Signals	Logic	Total	
16-bits	11	18	12	41	11.25
12-bits	10	11	7	28	7.68
10-bits	8	8	6	22	6.03
8-bits	6	6	5	17	4.66
7-bits	6	5	4	15	4.11
6-bits	5	4	3	12	3.29
5-bits	4	4	3	11	3.02

TABLE V
FPGA RESOURCE UTILIZATION FOR MNIST WITH DIFFERENT BITWIDTH WHILE USING DPR

Number of Bits	FF	LUT	Memory LUT	BRAM
16	14615	19141	2919	9
12	12511	15500	2471	9
10	11459	12646	2247	9
8	10407	10186	2023	9
7	9881	9329	1911	9
6	9355	8590	1799	9
5	7704	1687	8826	9
Available	106400	53200	17400	140

REFERENCES

[1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, ed.), pp. 396–404, Morgan-Kaufmann, 1990.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv:1602.02830*, 2016.

[4] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[5] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pp. 145–150, EDA Consortium, 2016.

[6] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2016.

[7] T. Ujiie, M. Hiromoto, and T. Sato, "Approximated prediction strategy for reducing power consumption of convolutional neural network processor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 52–58, 2016.

[8] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 475–480, 2016.

[9] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "Axnn: energy-efficient neuromorphic systems using approximate computing," in *Proceedings of the 2014 international symposium on Low power electronics and design*, pp. 27–32, ACM, 2014.

[10] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "Approxann: An approximate computing framework for artificial neural network," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 701–706, EDA Consortium, 2015.

[11] J. Chang and J. Sha, "An efficient implementation of 2d convolution in cnn," *IEICE Electronics Express*, pp. 13–20161134, 2016.

[12] J. Park and W. Sung, "Fpga based implementation of deep neural networks using on-chip memory only," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1011–1015, 2016.

[13] A. Hadnagy, B. Fehér, and T. Kovácsházy, "Efficient implementation of convolutional neural networks on fpga," in *2018 19th International Carpathian Control Conference (ICCC)*, pp. 359–364, 2018.

[14] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.

[15] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, vol. 181, p. 185, 1997.

[16] B. Moons and M. Verhelst, "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 237–242, 2015.

[17] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst, "Energy-efficient convnets through approximate computing," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8, 2016.

[18] XILINX, "UG909: Vivado Design Suite User Guide - Partial Reconfiguration (v2017.1)." Tech. Rep., 2017.

[19] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits." http://yann.lecun.com/exdb/mnist/, 2017.

[20] Keras, "https://keras.io/optimizers/," 2017.