Reconfigurable Activation Functions for Neural Networks Application

Yu-Jung Huang
Dept. of Electronic Engineering,
I-Shou University, Kaohsiung, Taiwan
email:
yjhuang@isu.edu.tw

Meng-Jhe Li
Dept. of Electronic Engineering
National Kaohsiung University of Science and Technology, Taiwan
email:
1106305149@nkust.edu.tw

Wun-Siou Jhong
Dept. of Electronic Engineering
National Kaohsiung University of Science and Technology, Taiwan
email:
f107152130@nkust.edu.tw

Shao-I Chu
Dept. of Electronic Engineering
National Kaohsiung University of Science and Technology, Taiwan
email:
erwinchu@nkust.edu.tw

Abstract - Field programmable gate arrays (FPGAs) have recently become popular for accelerating the deep learning networks due to their parallel processing and reconfigurable capabilities as well as their energy efficiency. This paper presents a multi-layer neural network architecture with novel reconfigurable activation functions by utilizing the coordinate rotation digital computer (CORDIC) technique and applying the floating-point format (IEEE 754 standard in single precision). The functionality was successfully verified in hardware using a DE2-115 board that included an Altera Cyclone® IV FPGA.

I. Introduction

An artificial neural network (ANN) is an interconnected group of nodes which perform functions collectively and in parallel, akin to human brain activities [1, 2]. ANNs have broad applicability to real world problems and are fast-growing artificial intelligence (AI) techniques used in industries during recent years. The artificial neuron represents real neuron mathematically. A large number of hardware architectures have been proposed for hardware implementation of ANNs. ANNs may be carried out by using analog systems or digital systems. In addition, existing platforms for hardware implementation of ANNs include digital signal processing (DSP) chips, application specific integrated circuits (ASICs), graphical processing unit (GPU) [3] and field programmable gate array (FPGAs) [4]. As the parallel structure of FPGAs matches the topologies of ANNs, they are quite suitable for the implementation of ANNs [5, 6].

The FPGA-based designs can accelerate the network classification process (forward computation) and achieve faster execution time and higher energy efficiency than CPU and GPU. This feature enables the neural network inference to have the advantage of reducing the cost of neural model development. Most FPGA implementations are reconfigurable by means of system regeneration and device reconfiguration to change the network topology. Although network training describes the problem of determining the parameters to model the target function, the activation function of the nodes can affect the training behavior of the network. Various activation functions lead to different convergence behavior and accuracy. For example, long short-term memory (LSTM) is the most commonly used model in the current recurrent neural network (RNN). The RNN is mainly applied to solve the problem of time series data. In training RNN, different kinds of activation functions can be applied to obtain the different shapes of the error surface. The presented reconfigurable activation functions can provide more flexible applications for neural networks. In this paper, the FPGA implementation with reconfiguration activation functions for neural networks is thus proposed.

II. Non-Linear Activation functions

A neuron forms the basis for designing the ANNs. The output activation f for the neuron is described by

$$o_i = f(\sum_{j=1}^{m} w_{ij} x_j + b_i). \quad (1)$$

w_{ij} denotes the weights connecting the jth input unit to the ith hidden unit. The weighted summation adds up the products of previous neurons multiplied by the corresponding weights, and then, the activation function is utilized to calculate the output. The bias b_i can be viewed as simply another weight (w_0) with a constant input of 1 (x_0=1). The role of activation functions is to make neural networks non-linear. The non-linearity takes a real-valued number and squashes it into the range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. The sigmoid function is frequently applied because it has a nice interpretation as the firing rate of a neuron. The ReLu function is fragile during the training procedure. A large gradient flowing through a ReLu-based neuron could cause the weights to update in such a way that the neuron will never activate on any data point again. By exploiting the MNIST dataset [7], an ANN model is trained by a two-layer perceptron to recognize the handwritten digits. Fig.1 depicts the training loss for two types of activation functions. For a two-layer perceptron, the training loss of the sigmoid function is less than that of ReLu function.

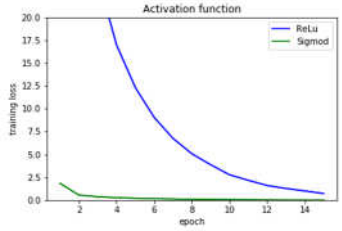
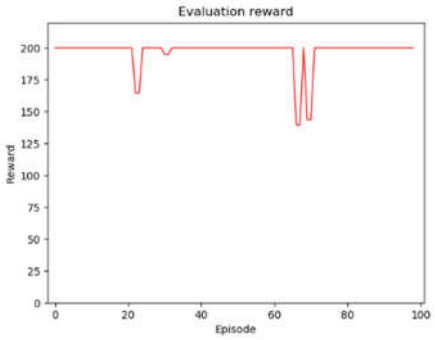


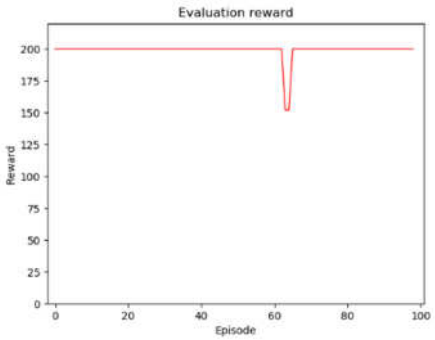
Fig. 1 Training loss over iterations for the ANN model

Fig. 2 shows the evaluation results of a neural network performance with different activation functions, including the sigmoid, ReLu and hypertangent (tanh) functions to train the

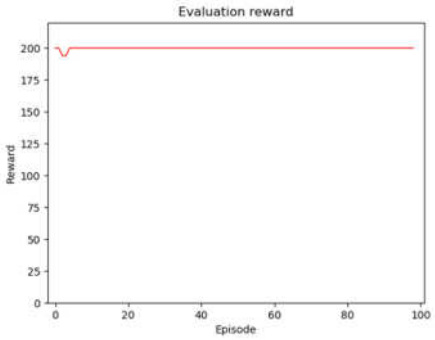
CartPole Agent in OpenAI gym [8]. It indicates that the tanh outperforms the other two activation function.



(a) Sigmoid function



(b) ReLu function



(c) Tanh function

Fig. 2 Learning curve for different activation functions

III. FPGA Implementation

Activation functions play an important role in the applications of deep learning and neural networks. A neural network with reconfigurable activation functions is implemented in the FPGA board. FPGA implementation of activation functions based on using the floating-point CORDIC technique [9] is shown in Fig. 3. The floating-point format is IEEE 754 standard in single precision. The CORDIC technique is a simple and efficient algorithm to implement the hyperbolic and trigonometric functions in hardware. We use the rotation mode to implement the exponential function.

As shown in Fig. 4, the Zynq SoC with an FPGA and an ARM core are adopted for our system implementation. AP SoC consists of an SoC-style integrated processing system (PS) and programmable logic (PL) on a single die. This evaluation board allows for full operation of the device to be investigated. A Vivado tool-generated IP block is integrated into a Zynq AP SoC. The Vivado tool-generated IP block is pre-verified on Altera Cyclone® IV FPGA.

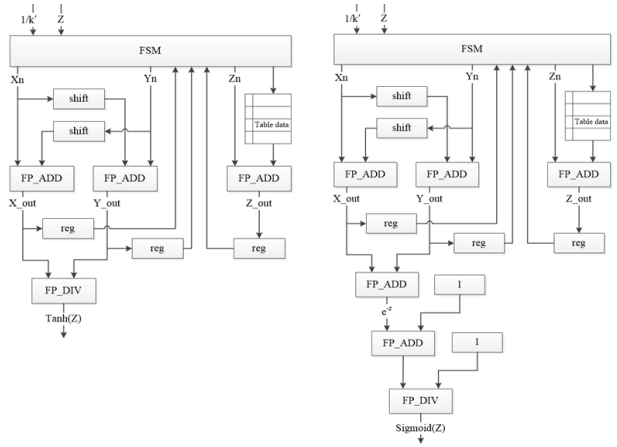


Fig. 3 FPGA implementation of activation functions based on the floating-point CORDIC technique

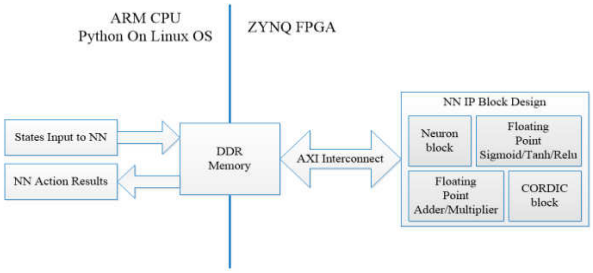


Fig. 4 System architecture of the proposed method

The implemented neural network is shown in Fig. 5, where the input for the neural network is (-0.3169452, -0.3536957, -0.003161569, 0.23862739) and the output results obtained are oxbf6ff9bb and ox3f54f7c1 by Python simulation.

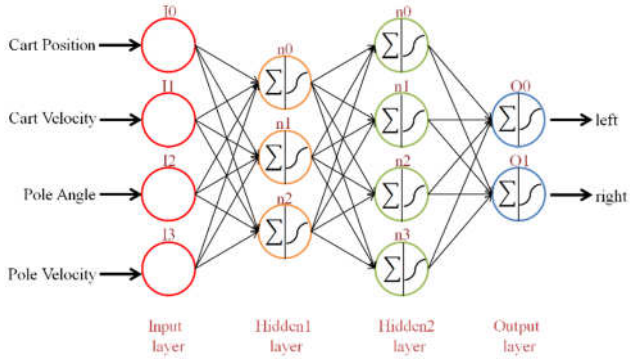


Fig.5.FPGA implementation of neural network with reconfigurable activation functions

In addition, the weights and the bias extracted from the trained neural network are listed in Table 1. The proposed design has been simulated and synthesized by using Synopsys design compiler logic synthesis tool. Table 2 summarizes the logic element gate counts and the power consumption based on the TSMC 130 nm technology file.

Table 1. Weights and bias extracted from the trained neural network

weight	Input 1	Input 2	Input 3	Input 4	bias
Hidden1 n0	3d97b032	bdc57f3b	bfac9f6c	bf5d4225	3e945d1c
Hidden1 n1	3d482b43	be398996	bfafca4c	bf598e34	3e99ca85
Hidden1 n2	bd38ab1c	3ec9ad78	3fa5ee4e	3f10b3d6	3eaf4ebc
	Hidden1 n0	Hidden1 n1	Hidden1 n2		
Hidden2 n0	3f3fcfe7	3f54a216	bf462ce2		3e34c2b6
Hidden2 n1	3f394b2b	3f4897dc	bf40a145		3e2fff6c
Hidden2 n2	bc94f551	bb07a8a3	bcd11dc4		bb372b22
Hidden2 n3	bc71eaa1	bc0f030f	bb2b5a14		00000000
	Hidden2 n0	Hidden2 n1	Hidden2 n2	Hidden2 n3	
Output 0	3f598f11	3f558439	3ce369c4	3ccdd157	bf5511d1
Output 1	bf59d8b9	bf4f79de	bcadea02	bcac0397	3f5511d1

Table 2 Design compiler report

Power Group	Internal	Switching	Leakage	Total	%
Register	40.1581	6.6978E-2	2.3126E+9	42.5385	88.49
Combinational	0.2173	1.9534	3.3609E+9	5.5311	11.51
Total (mW)	40.3754	2.0204	5.6735	48.0609	100.0
Combinational area	4436725.231794 um^2				
Buf/Inv area	672944.402659 um^2				
Non-combinational area	2589174.853397 um^2				
Total cell area	7025900.085191 um^2				

Table 3 lists the Quartus report based on Cyclone IV GX family EP4CGX150DF31C7 device, where the number of total logic elements is 135149. Simulation results of the ANN-based architecture using Verilog with the same inputs are shown in Fig. 6. It is observed that the output results are consistent with by Python simulations, listed in Table 4. It implies the successful and accurate implementation.

Table 3 Quartus report

Total logic elements	135149 / 149760 (90%)
Total registers	54158
Total pins	198 / 508 (39%)
Embedded Multiplier 9-bit elements	518 / 720 (72%)

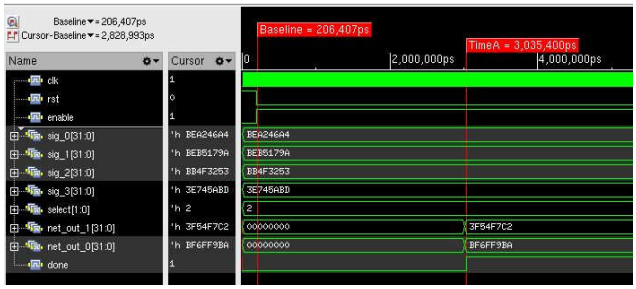


Fig. 6 Simulation result of the ANN with reconfigurable activation function

Table 4. Results of software and hardware

Env data	Python (value)	Modelsim (i754)	Modelsim (value)	sw-hw error
-0.3169452 -0.3536957 -0.0031616 0.2386274	Output0: -0.937404332 Output1: 0.831905435	bf6ff9ba 3f54f7c2	-0.9374043 0.8319055	3.2E-8 6.5E-8
-0.431019142 -0.338862689 -0.007135561 0.228214161	Output0: -0.645919531 Output1: 0.6483183988	bfa255fb 3f25f832	-0.6459195 0.6483184	3.1E-8 1.2E-9
-0.457088163 -0.143544401 0.003705012 -0.068809343	Output0: -0.000970211 Output1: 0.0117675211	ba7e5c00 3c40ccc0	-9.70304E-4 0.011767566	9.2E-8 4.4E-8
-0.478215659 -0.143402603 0.001295675 -0.071936886	Output0: 0.0161192181 Output1: -0.005099163	3c840c20 bba71580	0.016119063 -0.00509899	1.5E-7 1.6E-7

The verified Verilog code was downloaded on an Altera Cyclone® IV FPGA in the Altera DE2 board. This Altera DE2 board includes an Altera Cyclone® IV FPGA as well as various on-board components. The FPGA implementation and verification platform are shown in Fig. 7, which can be used simultaneously for comparison of the simulation and implementation results. In Fig. 8, the results of the FPGA implementation are further measured by the HP 16702A logic analyzer for real-time verification.

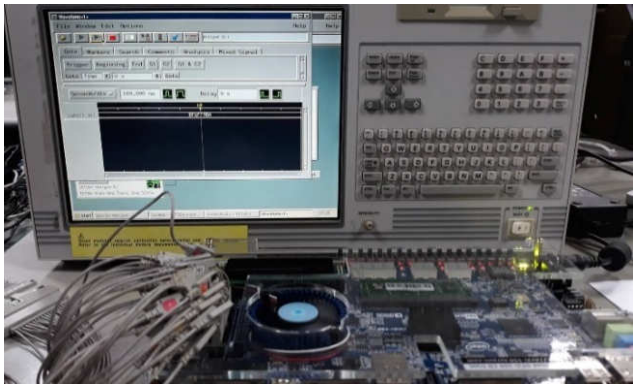


Fig. 7 FPGA implementation and verification platform

Fig. 8 Output obtained from a logic analyzer

V. Summary and Conclusions

Most of the neural networks have been applied for image applications. It might be enough using fixed-point precision for pixel-type data. However, as the neural network is applied for other applications such as cartpole problem, the floating-point computation becomes necessary to obtain the correct results. The proposed ANN architecture, which consists of reconfigurable activation functions with floating-point arithmetic, has been realized in the FPGA devices. Results reveal the successful implementation of the neural networks by using the CORDIC technique.

References

[1] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2ed. Addison Wesley Longman (Singapore) Private Limited, Delhi, 2001

[2] B. Schölkopf, "Artificial intelligence. Learning to see and act," *Nature*, vol.518, pp. 486–487, 2015.

[3] Shuoxin Lin, Yanzhou Liu, William Plishker and Shuvra S. Bhattacharyya, "A design framework for mapping

vectorized synchronous dataflow graphs onto cpu gpu platforms," *19th International Workshop on Software and Compilers for Embedded Systems*, pp. 20–29, New York, NY, USA, 2016. ACM.

[4] Z. Li, Y. J. Huang and W. C. Lin, "FPGA implementation of neuron block for artificial neural network," 2017 *International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pp. 1-2, 2017.

[5] A. Shawahna, S. M. Sait and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[6] S. I. Venieris, A. Kouris and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions," *ACM Computing Surveys*, vol. 51, no. 3, Jun. 2018.

[7] Y. LeCun, MNIST Handwritten Digit Database, [Online]. Available: http://yann.lecun.com/exdb/mnist/

[8] https://gym.openai.com/

[9] V. Tiwari and N. Khare, "Hardware implementation of neural network with sigmoidal activation functions using CORDIC," *Microprocessors and Microsystems*, vol. 39, no. 6, pp. 373-381,2015.