Implementation of Activation Functions using various approximation methods

Jiho Park, Geon Shin and Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Republic of Korea
jhpark.cas@gmail.com, gshin.cas@gmail.com, hyyoo@cnu.ac.kr

2024 21st International SoC Design Conference (ISOCC) | 979-8-3503-7708-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISOCC62682.2024.10762658

Abstract—**This paper compares the performance of activation function hardware under exponential function approximation techniques. The activation function is a key component of deep neural networks, allowing them to have nonlinear properties. Activation functions use exponential functions to obtain nonlinear properties, which make hardware implementation challenging. Exponential function hardware is implemented using representative approximation techniques such as Look-up-table, CORDIC, and Taylor series expansion. In addition, an approximation technique suitable for the activation function are suggested by performing an analysis of the hardware operation accuracy and area on the activation functions Sigmoid, Tanh, Swish, and GELU. Synthesis result shows CORDIC is area-efficient than other techniques in all functions. For accuracy, the Taylor series expansion in Tanh is the best, and CORDIC is good in the other activation functions.**

Keywords; Deep neural network, Activation function, Look-Up-Table, CORDIC, Taylor series.

I. INTRODUCTION

Deep neural networks are being used to solve problems in many fields, such as image classification, speech recognition, and language models [1]. A deep neural network refers to a neural network with more than one hidden layer, and it allows nonlinear relationships between layers to solve complex problems. Recently, new methods of activation functions have been studied to enhance the performance of deep neural networks. Sigmoid and tanh are used in many neural networks because they are differentiable and are compatible with backpropagation algorithms. In addition, activation functions such as swish and GELU have been proposed to solve problems such as gradient loss [2], and GELU is being used in large language models such as GPT-3 and BERT.

Exponential functions present in activation functions make the hardware implementation of activation functions difficult. Approximation methods are generally used to implement nonlinear function. Look-up-table (LUT) [3], CORDIC [4], and Taylor series expansion [5] approximation methods are typically used. In this paper, the activation function is implemented using three representative approximation methods. After that, we compare the performance of the implemented hardware and examine what is an efficient approximation method for each activation function.

II. ACTIVATION FUNCTION

Sigmoid is the most common activation function and is used in many neural networks. When the absolute value of the input value is large, the output value is saturated and the performance of learning is degraded due to gradient vanishing.

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}. \quad (1)$$

Tanh is an activation function that shows better performance than the sigmoid function when there are many layers in neural networks. Like sigmoid, there are difficulties due to gradient vanishing.

$$\text{Tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \quad (2)$$

Swish is an activation function containing learnable parameters. β is a learnable parameter, and when this value is 1, it is equal to the SiLU activation function. The value of β increases, it becomes the same as the ReLU function.

$$\text{Swish}(x) = \frac{1}{1+e^{-\beta x}} \cdot x. \quad (3)$$

A Gaussian error linear unit (GELU) is an activation function based on a standard Gaussian cumulative distribution. Due to the complexity of the formula, it is approximated by the following equation. GELU shows better performance than conventional activation function.

$$\text{GELU}(x) \approx \frac{1}{2}x\left(1+\text{Tanh}\left[\sqrt{2/\pi}(x+0.044715x^{3})\right]\right). \quad (4)$$

III. HARDWARE IMPLEMENTATION

In this paper, the exponential function is implemented using LUT, CORDIC, Taylor series expansion approximation methods. Based on this, an hardware was designed to calculate Sigmoid, Tanh, Swish, GELU functionis The number notation method for the implementation of the activation function hardware has a 2's complement, using a 32-bit fixed-point method, and 12 bits for the exponential part and 19 bits for the mantissa part.

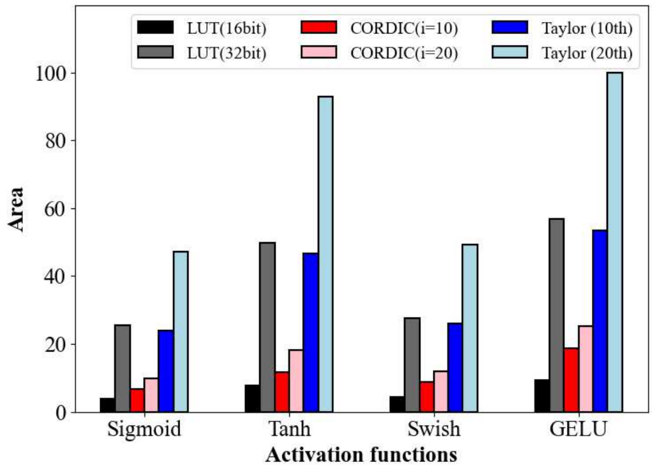

Figure 1. Hardware area for various activation functions

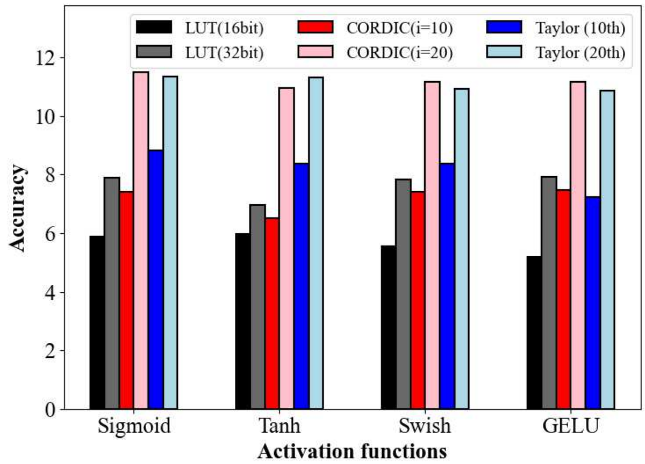

Figure 2. Accuracy for various activation functions

The LUT method uses a table that stores the result value calculated in advance for the input of an exponential function. A table is created by calculating all the results for each bit of the input. When an input comes in, the operation results for each bit are taken from the table and multiplied to output the exponential function operation result for the input.

The CORDIC method obtains the operation result of the trigonometric function using the rotation vector and calculates the exponential function result using this value. The formula of the CORDIC algorithm for exponential function shown as

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}, y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \mathrm{Tanh}^{-1}\left(2^{-i}\right) \quad . \tag{5}$$
$$d_i = +1\left(x_i < 0\right), d_i = -1\left(x_i \geq 0\right)$$

Value of x, y is the vector position, z is input angle, d is the rotation direction, i is the number of rotation iterations. Initial values for obtaining exponential function results are x_0 = 1/0.8281, $y_0 = 0$, and z_0 is an input value.

The Taylor series expansion method approximates the exponential function to a polynomial. The larger the order of the polynomial, the higher the accuracy. The approximation equation of the exponential function is as follows

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \ . \tag{6}$$

The exponential function can be calculated only by addition and multiplication.

IV. EXPERIMENTAL RESULT AND CONCLUSION

The performance is compared through hardware area and accuracy. The area of the hardware is obtained through synthesis. The synthesis environment used a 28 nm CMOS process, the target frequency is 500 MHz, and implementation synthesized through Synopsys Design Compiler S-2021.06. Fig. 1 shows the area of hardware that implements the various activation functions mentioned above.

Accuracy is calculated through the mean square error (MSE) and is shown as

$$\text{accuracy} = \log\left(\frac{1}{MSE}\right), MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2 \ . \tag{7}$$

Value of y is Python simulation, t is hardware simulation result, and $n = 1000$. Fig. 2 shows the accuracy of activation function.

As a result of comparing the hardware performance of the approximation method, the CORDIC method is the most area-efficient in all activation function and shows high accuracy. The Taylor series expansion method also shows high accuracy, but has the disadvantage of large area. The LUT method shows low area efficiency and accuracy, but has the advantage of being easy to design hardware. disadvantage of large area. Through the comparison results, this paper proposes a method suitable for the implementation environment.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A5A8026986), supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01170), and the EDA tool was supported by the IC Design Education Center (IDEC), Korea.

REFERENCES

[1] S. Drimer, “Volatile FPGA design security–a survey,” Computer Laboratory, University of Cambridge, 2008.

[2] Apicella, Andrea, et al. "A survey on modern trainable activation functions." Neural Networks 138 (2021): 14-32.

[3] Xie, Yusheng, et al. "A twofold lookup table architecture for efficient approximation of activation functions." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 28.12 (2020): 2540-2550.

[4] Heidarpur, Moslem, et al. "CORDIC-SNN: On-FPGA STDP learning with izhikevich neurons." IEEE Transactions on Circuits and Systems I: Regular Papers 66.7 (2019): 2651-2661.

[5] Nilsson, Peter, et al. "Hardware implementation of the exponential function using Taylor series." 2014 NORCHIP. IEEE, 2014.