Neural Networks Probability-Based PWL Sigmoid Function Approximation

Vantruong NGUYEN†, Jueping CAI†(a), Linyu WEI†, Nonmembers, and Jie CHU†, Member

SUMMARY In this letter, a piecewise linear (PWL) sigmoid function approximation based on the statistical distribution probability of the neurons’ values in each layer is proposed to improve the network recognition accuracy with only addition circuit. The sigmoid function is first divided into three fixed regions, and then according to the neurons’ values distribution probability, the curve in each region is segmented into sub-regions to reduce the approximation error and improve the recognition accuracy. Experiments performed on Xilinx’s FPGA-XC7A200T for MNIST and CIFAR-10 datasets show that the proposed method achieves 97.45% recognition accuracy in DNN, 98.42% in CNN on MNIST dataset, and 72.22% on CIFAR-10 dataset, respectively, up to 0.84%, 0.57% and 2.01% higher than other approximation methods with only addition circuit.

key words: sigmoid function, probability, neural networks, piecewise linear approximation

1. Introduction

Artificial Neural Network (ANN) is a multi-layer neural network structure, in which each layer has over hundreds of neurons, and each neuron value is calculated by multiplication, addition and activation operation. The sigmoid function is a common activation function in ANNs, and it is a challenge to implement in hardware directly, because it requires a large amount of resources to implement exponentiation and division calculation [1].

Many approximation methods such as look-up tables, CORDIC algorithm, Taylor series expansion, polynomial, piecewise methods [2]–[8] are presented to simplify the hardware implementation complexity, but the accuracy decreases as the complexity decreases.

In order to achieve high accuracy sigmoid function approximation in the condition of low complexity hardware implementation, a new PWL approximation based on the neurons’ values distribution probability is proposed in this letter. The curve of the sigmoid function is divided into three fixed regions firstly, and then according to the neurons’ values distribution probability in each region, the region with the higher probability is segmented sub-regions to reduce the approximation error. Finally, three PWL approximation functions with the slope of $2^{-n}$ used in the different layers are proposed to improve the recognition accuracy, while the hardware complexity is reduced with only addition and shifting computation. The implementation results performed on FPGA-XC7A200T for MNIST and CIFAR-10 datasets show that the proposed method achieves the recognition accuracy of 97.45% in DNN, 98.42% in CNN on MNIST dataset, and 72.22% on CIFAR-10 dataset, respectively, up to 0.84%, 0.57% and 2.01% higher than the approximation method with only addition circuit.

2. Proposed Probability-Based PWL Approximation

2.1 Three Fixed Regions Segmentation

The curve of the sigmoid function defined by Eq. (1), and its first four derivatives are shown in Fig. 1, in which $f(x)$ varies in the range (0, 1) with the different slopes.

\[ f(x) = \frac{1}{1 + e^{-x}} \tag{1} \]

According to the variance rate of slope described by the sigmoid second derivative (as Eq. (2)), the sigmoid function based on the X-axis is divided into three fixed regions, including the approximate linear region (region I) which the variance rate of the slope is fast, the saturation region (region II) which the variance rate of the slope is slower, and the approximation constant region (region III) which is the rest of sigmoid function (shown in Fig. 1).

\[ f''(x) = \frac{e^{-2x} - e^{-x}}{(1 + e^{-x})^3} \tag{2} \]

The demarcation point between the approximation linear region and saturation region is the inflection point of the sigmoid second derivative which segments sigmoid function into the concave and convex region. This point $x$ is 2.2 calculated by taking the fourth derivative (as Eq. (3)) equals to zero.

\[ f^{(4)}(x) = -e^{-x} + 11e^{-2x} - 11e^{-3x} + e^{-4x} \left(1 + e^{-x}\right)^2 \] \tag{3}

![Fig. 1 Sigmoid function and its first fourth derivatives](image-url)
The start point of approximation constant region is determined by the maximum allowable error \( \Delta \) between the sigmoid function and “1” function which is calculated by Eq. (4), in which \( \lfloor \cdot \rfloor \) is the integer from the floating-point number.

\[
x_{\text{cont}} = \left\lfloor \ln \left( \frac{1 - \Delta}{\Delta} \right) \right\rfloor
\]

The maximum allowable error is set to 0.005, and the corresponding starting point of the approximation constant region is 5. Hence, the approximation linear region, saturation region, and approximation constant region in the positive X-axis are [0, 2.2], [2.2, 5] and [5, \( \infty \)], respectively.

\[
f(-x) = 1 - f(x)
\]

Because the sigmoid function is symmetric at point \((0, 0.5)\), when the function value \( f(x) \) of the positive axis is calculated, the function value \( f(-x) \) of the negative axis is calculated by Eq. (5). The approximation linear region, saturation region, and approximation constant region in the negative X-axis are \([-2.2, 0], [-5, -2.2] \) and \((\infty, -5)\).

### 2.2 Probability-Based Sub-Regions Segmentation

The neurons’ values distribution probability in the above three fixed regions is different in each layer. As shown in Fig. 2, assuming that there are \( M \) layers in ANN, and \( N \) neurons in the \( l^{th} \) layer. The neurons’ values distribution probability of the \( l^{th} \) layer in region \( i \) is given by

\[
P_i^l = \frac{K_i^l}{N}
\]

where \( K_i^l \) is the number of neurons in the \( l^{th} \) layer in which the neuron’s value belongs to region \( I (i = I, II, III) \).

To guarantee an even number of segments per half axis, the total number of segments is a multiple of 4. Experiments have been verified with the total segments number of 8, 12 and 16 in the range of \([-5, 5]\), and 12 segments number is selected based on the overall performance. When the total segments are determined, the number of sub-segments in each region is determined according to the neurons’ values distribution probability which means there are more sub-segments in the higher probability region.

The sigmoid function can be approximated by an \( n \)-order Taylor polynomial as Eq. (7), in which the approximation error \( R_n(x) \) is defined by the Lagrange remainder term. Equation (8) shows the \((n + 1)\)-order derivative directly affects the approximation error at the point when using a degree \( n \)-order polynomial to fit the sigmoid function. If the point where the absolute value of \((n + 1)\)-order derivative is larger, the approximation error is larger. To reduce the approximation error, the smaller width sub-regions are segmented near the point where the absolute value of the \((n + 1)\)-order derivative is larger.

\[
f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x - a)^n + R_n(x)
\]

\[
R_n(x) = f^{(n+1)}[a + \theta(x-a)] (x-a)^{n+1} \frac{1}{(n+1)!}, \ \theta \in (0, 1)
\]

Because the basic PWL function used to fit the sigmoid function, defined by Eq. (9), is the first-order function in which \( m \) is the function slope, and \( b \) is the constant. According to the property of Taylor’s formula, the width of the sub-regions in this paper is determined by the second derivative of the sigmoid function.

\[
f(x) = mx + b
\]

#### 2.3 Proposed Probability-Based PWL Functions

From Eq. (7), when \( a \) is zero, the PWL sigmoid approximation function \( f(x) = 0.25x + 0.5 \) with the slope \( m = 2^{-2} \) which approximates the sigmoid function better by converting the multiplication into the shifting when \( x \) is close to the zero. As a result, the slope of \( 2^{-n} \) is chosen in this paper to reduce the hardware implementation complexity with only addition circuit when realizing the PWL function in hardware. The PWL function in the range \( [a_i, c_i] \) given by

\[
\hat{f}(x) = \frac{x}{2^n} + b_i
\]

where \( n \) and \( b_i \) minimize the maximum absolute error \( E_{\text{abs}} \) between the PWL function and sigmoid function which is

\[
E_{\text{abs}} = \left| \max(\hat{f}(x) - f(x)) \right|
\]

Table 1 shows the parameters \( n, b_i \) of the three proposed PWL approximation function at the positive X-axis approximation linear region and saturation region.

| Range     | \( n \) | \( b_i \) | Range     | \( n \) | \( b_i \) | Range     | \( n \) | \( b_i \) |
|-----------|--------|--------|-----------|--------|--------|-----------|--------|--------|
| [0, 0.1)  | 2      | 0.4877 | [0.1, 0.5)| 2      | 0.4906 | [0.5, 0.9]| 2      | 0.4931 |
| [1, 1.2)  | 3      | 0.6231 | [1.5, 2   | 3      | 0.6182 | [0, 1)   | 3      | 0.6110 |
| [2, 2.2)  | 4      | 0.7657 | [2.3, 3   | 3      | 0.6293 | [0, 0.9)| 3      | 0.6293 |
| [2.6, 3.2)| 4      | 0.7674 | [2.2, 5   | 3      | 0.7657 | [1, 2]   | 3      | 0.6293 |
| [3, 3.2)  | 5      | 0.8586 | [2.3, 6   | 4      | 0.7647 | [2.2, 3.6| 4      | 0.7583 |
| [4, 4.5)  | 7      | 0.9331 | [3.2, 10] | 6      | 0.9153 | [2.6, 5] | 6      | 0.9175 |

**Fig. 2** Basic structure of ANNs
3. Implementation Results

3.1 Approximation Performance

Figure 3 shows the curves of three proposed PWL functions, in which the maximum absolute error is low in the high probability region, and large in the low probability region. As shown in Table 2, the proposed method reduces the maximum and average absolute error to 0.0125 and 0.0042.

3.2 Recognition Accuracy

Experiments are implemented on FPGA-XC7A200T for recognizing MNIST and CIFAR-10 datasets in the different ANN networks to evaluate the recognition performance. The architectures and the neurons’ values distributions of DNN and CNNs are shown in Tables 3, 4 and 5.

Table 6 shows the computation of activation approximation methods in different CNNs for different datasets.

Table 7 shows the recognition accuracies on MNIST and CIFAR-10 datasets. The proposed method achieves high recognition accuracy of 97.45% in DNN and 98.42% in CNN on MNIST, and 72.22% in CNN on CIFAR-10 dataset, up to 0.84%, 0.57% and 2.01% higher than the approximation method of the reference [3].

Due to the approximation error, the neurons’ value is changed (increases or decreases) which affects the neurons’ value of the next layer. The error accumulated in the whole network leads to the recognition result to be changed when the two neurons’ values in the output layer are very close. Figure 4 shows four examples for the different recognition results between our proposed approximation method and the sigmoid function, in which our proposed PWL method recognizes to “8”, the same as the images labels, but the original sigmoid function recognizes to “6”, “1”, “2” and “3”.

Table 2

| Method       | Range   | Piecewise number | Absolute Error |
|--------------|---------|------------------|----------------|
| Nghi [6]     | [-4,4]  | -                | 0.022 0.0077   |
| Gomar [5]    | [-4,4]  | -                | 0.0087 0.0058  |
| Mitta [7]    | [-9.35,35] | 14               | 0.0127 0.0015  |
| Zamanloooy [8]| [-8,8]  | 6                | 0.0189 0.0059  |
| Campo [2]    | [-4.59,4.59] | 12              | 0.0280 0.0043  |
| Savich [3]   | [-8,8]  | 5                | 0.0679 0.0265  |
| Proposed method | [-5.5] | 12               | 0.0125 0.0042  |

Table 3

| Layers        | Neurons number | Probability of neurons’ values (%) | Region I | Region II | Region III |
|---------------|----------------|------------------------------------|----------|-----------|------------|
| Input         | 784            |                                    | -        | -         | -          |
| Hidden1       | 576            | 54.69                              | 43.93    | 1.38      |            |
| Hidden2       | 450            | 98.96                              | 1.04     | 0         |            |
| Hidden3       | 300            | 92.63                              | 7.37     | 0         |            |
| Hidden4       | 120            | 42.25                              | 57.5     | 0.25      |            |
| Hidden5       | 80             | 26.29                              | 71.45    | 2.26      |            |
| Output        | 10             | 0.8                                | 11       | 88.2      |            |

Table 4

| Layers        | Size       | Probability of neurons’ values (%) | Region I | Region II | Region III |
|---------------|------------|------------------------------------|----------|-----------|------------|
| Input         | 1x32x32    |                                    | -        | -         | -          |
| Conv_1        | 6x32x28    | 63.15                              | 32.23    | 4.62      |            |
| Pool_1        | 6x14x14    |                                    | -        | -         | -          |
| Conv_2        | 12x10x10   | 26.46                              | 58.27    | 15.27     |            |
| Pool_2        | 12x5x5     |                                    | -        | -         | -          |
| FullConv_1    | 120        | 53.09                              | 46.77    | 0.14      |            |
| FullConv_2    | 10         | 2.2                                | 10.72    | 87.08     |            |

Table 5

| Layers        | Size       | Probability of neurons’ values (%) | Region I | Region II | Region III |
|---------------|------------|------------------------------------|----------|-----------|------------|
| Input         | 3x32x32    |                                    | -        | -         | -          |
| Conv_1        | 32x30x30   | 53.46                              | 39.15    | 7.39      |            |
| Pool_1        | 32x14x14   | 41.68                              | 47.90    | 10.42     |            |
| Conv_2        | 64x12x12   | 25.31                              | 40.12    | 34.57     |            |
| Conv_3        | 64x10x10   | 28.75                              | 37.53    | 33.72     |            |
| Pool_2        | 64x5x5     |                                    | -        | -         | -          |
| Conv_5        | 128x3x3    | 44.60                              | 38.19    | 17.21     |            |
| Conv_6        | 128x1x1    | 37.02                              | 43.45    | 19.53     |            |
| FullConv_1    | 84         | 66.17                              | 28.63    | 5.20      |            |
| FullConv_2    | 10         | 15.61                              | 31.09    | 53.30     |            |

Table 6

| Method        | MNIST/CIFAR | MNIST/CIFAR | MNIST/CIFAR |
|---------------|-------------|-------------|-------------|
| Nghi [6]      | 12068/141756 | 12068/141756 | 6034/70878  |
| Gomar [5]     | 0/0        | 12068/141756 | 6034/70878  |
| Campo [2]     | 6034/70878  | 6034/70878  | 0/0         |
| Proposed method | 6034/70878  | 6034/70878  | 0/0         |

Table 7

| Method        | MNIST/CIFAR |
|---------------|-------------|
| Nghi [6]      | 97.37/98.96 | 97.37/98.35 | 72.16       |
| Gomar [5]     | 97.30/98.24 | 97.30/98.24 | 72.04       |
| Mitta [7]     | 97.35/98.29 | 97.35/98.29 | 72.15       |
| Zamanloooy [8]| 97.36/98.21 | 97.36/98.21 | 72.11       |
| Campo [2]     | 97.34/98.27 | 97.34/98.27 | 72.13       |
| Savich [3]    | 96.61/97.85 | 96.61/97.85 | 70.21       |
| Average PWL   | 97.38/98.34 | 97.38/98.34 | 72.13       |
| Proposed method | 97.45/98.42 | 97.45/98.42 | 72.22       |
3.3 Hardware Implementation

The circuit architecture of the proposed probability-based PWL approximation implemented on FPGA is shown in Fig. 5, in which the multiplication is replaced by shifting. Considering the accuracy, computational complexity, and hardware cost, the 16-bit fixed point (1 symbol bit, 3 integer bits and 12 fractional bits) is used to represent the weights and the inputs X of the activation function, while the activation function output Y use the 13-bit fixed point with 1 integer bit and 12 fractional bits.

Table 8 shows the hardware usage of the proposed PWL approximation function and the neural network system implemented on a Xilinx Artix-7 (XC7A200T) with the working frequency is 140MHz. The results show that the resource utilization of the proposed function is much low with none DSP is used. The system achieves 35.16 Giga operations per second at the frequency of 140MHz.

4. Conclusion

A Probability-based PWL sigmoid function approximation method is proposed to guarantee the low complexity of hardware implementation for achieving high recognition accuracy. The method provides a piecewise way based on the neurons’ value distribution probability to reduce approximation error and improve recognition accuracy with only addition circuit. Experiments performed on FPGA implementing the MNIST and CIFAR-10 datasets. The proposed method achieves high recognition accuracy of 97.45% in DNN on MNIST dataset, 98.42% and 72.22% in CNN CIFAR-10 dataset with only addition circuits, which are 0.84%, 0.57%, and 2.01% higher than other methods with the same computation complexity.

Acknowledgments

This work was supported by the innovation wisdom base for wide bandgap semiconductor and micro-nano electronics of China (B12026), Wuhu and Xidian University special fund for industry-university-research cooperation, China (0120 19005), Strategic international scientific and technological innovation cooperation key projects (2016YFE0207000).

References

[1] Z. Hajduk, “Hardware implementation of hyperbolic tangent and sigmoid activation function,” Bulletin of the Polish Academy of Sciences, Technical Sciences, vol.66, no.5, pp.563–577, 2018.

[2] I. Campo, R. Finker, J. Echanobe, and K. Basterretxea, “Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons,” Electronics Letters, vol.49, no.23, pp.1598–1600, 2013.

[3] A.W. Savich, M. Moussa, and S. Areibi, “The impact of arithmetic representation on implementing MLP-BP on FPGAs: A Study,” IEEE Trans. Neural Netw., vol.18, no.1, pp.240–252, 2007.

[4] T. Sasao, S. Nagayama, and J.T. Butler, “Numerical Function Generators Using LUT Cascades,” IEEE Trans. Comput., vol.56, no.6, pp.826–838, 2007.

[5] S. Gomar, M. Mirhassani, and M. Ahmadi, “Precise digital implementations of hyperbolic tanh and sigmoid function,” 2016 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, pp.1586–1589, 2016.

[6] S. Ngah and R.A. Bakar, “Sigmoid function implementation using the unequal segmentation of differential lookup table and second order nonlinear function,” Journal of Telecommunication, Electronic and Computer Engineering, vol.9, no.2, pp.103–108, 2017.

[7] S. Mitra and P. Chattopadhyay, “Challenges in implementation of ANN in embedded system,” 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, pp.1794–1798, March 2016.

[8] B. Zamanlooy and M. Mirhassani, “An analog cvns-based sigmoid neuron for precise neurochips,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.25, no.3, pp.894–906, March 2017.