**2DAN-BNN: Two-Dimensional AN-Code Decoders for Binarized Neural Networks**

*Kuan-Yu Huang1, Jettae Schroff 2, Cheng-Di Tsai1 and Tsung-Chu Huang1,*

*1*Department of Electronics Engineering, National Changhua University of Education, Taiwan ROC

*2*Cogito Academy, Orinda, California US

*Abstract*—Binarized neural network (BNN) becomes the low-cost high-accelerated inference machine for consumer electronics. But for some safety-critical applications, reliability is more critical than cost and acceleration. AN codes are effective and efficient for improving the reliability of a BNN, but each class needs a decoder. In this paper, we propose a two-dimensional error-locating technique for sharing only a decoder. From experimental results, the logic elements can be highly reduced up to 93%.

# Introduction

*Consumer electronics* with artificial intelligence (AI) have strongly speeded up computing acceleration up to twice per 3.4 months [1]. Owing to high acceleration with low power and low cost in security-critical electronics, on one hand quantization and even binarization has widely applied in a neural network (NN) [2]. On the other hand, this results in another critical issue, reliability. Although input-side layers of a neural network (NN) have been simulated and analyzed to have a high self-healing ability [3], any bit of computing errors near output-side layers will still cause a wrong decision drastically, and even a catastrophe.

Quantization NN (QNN) is a direct and effective accelerating methodology for reducing power dissipation and hardware cost [4]. In a W*w*A*a*-modeled QNN, the weights and activation functions are fixed in *w* and *a* bits. The binarized NN (BNN), W1A1 even minimized the number lengths to the extreme, in which the weights will be binarized to either or for propagation except gradients and probabilities still with longer fixed-point fractions [5]. Fig.1(a) shows the synapses of a output node in a typical layer of the trained count-based BNNs, where each count of input fan-out synapses is equivalent to its weight, therefore they need a lot of input branches. In [6], TWBNN took {+1, 0, -1} as the weights, but the trained NN will be still a count-based BNN. To reduce the synapses, the authors in TWBNN converted the count to ternary-coded binary (TCB) as shown in Fig.1(b) using physical shift registers [7]. In AN-TCBNN [8], the authors transferred the shift registers to constant shifts for saving area and firstly applied the product (AN) codes for correcting a single arithmetic weight error (AWE) [9]. However, their decoder takes a big look-up table (LUT) circuitry and the whole NN requires decoders for the class nodes in the *L*th layer. Therefore in this paper, we propose a two-dimensional error location technique for sharing only one AN decoder (2DAN) for BNNs.

(a) (b)

Fig. 1 (a) Count-based and (b) shift-based BNNs.

The rest of this article is organized as follows. In Sec. II, the AW minimization algorithm and the AN Codes are reviewed. The proposed 2DAN is then presented in Sec. III. In Sec. IV, experimental and evaluated results are presented. Finally, the conclusions are withdrawn in Sec. V.

# Definitions and Preliminary Reviews

## Arithmetic Weight and Arithmetic Weight Error Model

An *n*-bit 2’s complemented binary-coded binary (BCB) number *x* can represented as

, (1)

where. It can be also represented by a TCB number,

, (2)

where and signs ± separately represent ±1. Here will be called a *digital weight* (DW) for distinguishing with NN *weights* and *arithmetic weight* (AW) of *x*. AW is defined as the minimum count of ± signs for representing, namely,

. (3)

The AW distance of any two integers *x* and *y*, is defined as . For convenience of explanation, a number with a AW no more than 2 (3) is called a nice (light) number respectively. The AW can be minimized by the AW-minimization algorithm [8].

## AN Codes

In a () AN codes, the message word *N*, , is encoded to the product codeword *C*=*AN*, possibly infected by an AWE, as , and finally decoded by . where . As illustrated in Fig.2, the *w*-AWE, *e*, can be located as , where denotes the residue of *x* mod *A*. Note that in the AWE model a single error can be or at bit location *i*, For example, a checkbits locates an error at index 3 of a Hamming codeword, while a residue (mod 37) denotes an error at location 3 of the 37N codes.



Fig. 2 Illustration of error location for AN and Hamming Codes.

If *A* is an odd prime number, and *2* is a primitive element for generating a multiplicative group G(*A*) of Galois field, GF(*A*), then G(*A*) is perfect to locate the AWE indices of a -bit *A*N codeword. Since it is easy for today’s computer, we will choose a good multiplier number A for AN codes by an exhaustive searching scheme without too many complicated theories for applications only. Some searched AN-code multipliers are shown in Table I, where Row ***w***(A) enumerates the AW of A. For example, 37 is a good multiplier for a perfect AN codes with single-error correcting (SEC) capability of a 18-bit codeword for , since GF(37)={0; (1, 2, 4, 8, 16, 32, 27, 17, 34, 31, 25, 13, 26, 15, 30, 23, 9, 18, 36, 35, 33, 29, 21, 5, 10, 20, 3, 6, 12, 24, 11, 22, 7, 14, 28, 19)}.

1. Multiplier examples for perfect AN codes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *n* | 5 | 6 | 9 | 11 | 14 | 18 | 23 | 26 | 29 | 30 | 33 | 35 | 39 | 41 | 50 | 51 | 53 | 55 | 65 |
| A | 11 | 13 | 19 | 23 | 29 | 37 | 47 | 53 | 59 | 61 | 67 | 71 | 79 | 83 | 101 | 103 | 107 | 121 | 131 |
| ***w***(A) | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 3 |

# Proposed 2DAN-BNN

## Activation Function

ReLU as shown in Fig.3(a) is *AN*-preserving since the output is either 0 or *x*, which is an *A*-multiple. However, the dynamic range (DR) will be accumulated through the later layers. In this paper, the piecewise linear unit (PLU) is modified and called Clipper as shown in Fig.3(b), where the output will be only within [10]. Therefore the Clipper is AN-preserving and able to limit the DR accumulation.

(a)  (b) 

Fig. 3 (a) a ReLU, and (b) a PLU activations.

## Proposed AN Decoder

Without loss of generality, the *AN* decoder can be explained using 13N codes. Fig.4(a) shows the error-quote-remainder table, where *AN + e = AN + AΔQ + R*. Then a 13*N* decoder can be designed as shown in Fig.4(b). Note that the division is highly reduced by the Barret algorithm. Furthermore, *A* is searched by minimizing the weight of , therefore the multiplication can be further implemented by three or four-input addition.

(a)  (b)

Fig. 4 (a) Error-Q-R table, and (b) A decoder of 13N codes.

## Two-dimension Detecting Error Location for AN Codes

Without loss of generality, assume the node count of the last *AN*-preserving layer is . Then each row/column error indicators can be implemented by an OR gate with inputs from all remainder detectors in the row/column as shown in Fig.4(a). Note that the quotients and remainders can be computed by the Barrett reduction. Then the error at node can be detected by an AND gate with inputs and , which is used to select remainderfor finding the complement number in the LUT for complementing the quotient as shown in Fig.5(b). Since the LUT dominates the area overhead of an AN decoder, the 2DAN error locating technique can highly reduce the area overhead.

(a)  (b) 

Fig. 5 (a) 2DAN Error location, (b) one decoder shared by outputs.

# Evaluation and Experimental Results

A -output DNN with all PLU activation functions is trained in workstations. Then the precision model is converted to 2DAN-BNN for only inferencing. In this paper, only the area and power improvement of the AN decoders are evaluated. The structures in Fig.4(b) and Fig.5 are implemented in Xilinx Vivado/Pynq-z2 for experiments. Table II shows the required counts of components in Xilinx z7020 for *k*=4, 5, and 6 separately when *A* = 13, 29, and 37.

1. Comparison of AN-Decoders with Previous Work.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Work | | Separate Decoders [8] | | | Ours | | |
| Components  in z7020 | | LUT | MUXFx | Carry  Logic | LUT | MUXFx | Carry  Logic |
|  | A=13 | 24,720 | 80 | 6,240 | 2,091 | 3 | 438 |
| A=29 | 60,848 | 1,152 | 13,312 | 7,372 | 104 | 1,417 |
| A=37 | 161,200 | 1,248 | 37,504 | 12,900 | 234 | 2,724 |
|  | A=13 | 38,600 | 125 | 9,750 | 2,398 | 2 | 465 |
| A=29 | 95,075 | 1,800 | 20,800 | 8,083 | 104 | 1,480 |
| A=37 | 251,875 | 1,950 | 58,600 | 14,254 | 312 | 2,949 |
|  | A=13 | 55,584 | 180 | 14,040 | 2,666 | 1 | 498 |
| A=29 | 211,104 | 4,464 | 47,232 | 8,987 | 104 | 1,557 |
| A=37 | 362,700 | 2,808 | 84,384 | 15,950 | 78 | 3,224 |

We find that no DSP is required when the dividers in [8] are implemented by modified Barret reduction using several-input adders with constant shifts. When the 2D error location technique is applied for sharing only a decoder, more than 93% of area can be saved.

# Conclusions

In this paper, we propose a 2D error location technique for help any AN-coded BNN to reduce the AN-decoders from ***O***(*k*2) to only 1. We also apply the PLU functions as the inter-layer activation functions for both AN-preserving and DR-clipping. This help the AN-encoders can be embedded in some hidden layer, and go through the last layers, and improve the area overhead and thus the associated power dissipation in previous work [8]. From our preliminary evaluation, the improvement can be up to 93% for many conventional BNNs.

##### References

1. OpenAI. AI and Compute. https://openai.com/blog/ai-and-compute. May 16, 2018.
2. S. Haykin. Neural Networks and Learning Machines. 3rd edition, Prentice Hall, 2009.
3. A. Bosio, P. Bernardi, A. Ruospo and E. Sanchez, "A Reliability Analysis of a Deep Neural Network," 2019 IEEE Latin American Test Symposium (LATS), Santiago, Chile, 2019, pp. 1-6.
4. I. Hubara *et al*. “Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations”, *arXiv e-prints*, 2016.
5. M. Courbariaux *et al*. “Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1”, *arXiv e-prints*, 2016.
6. Li, F. and Liu, B. Ternary weight networks. CoRR, abs/1605.04711, 2016.
7. R. Ding, Z. Liu, R. D. S. Blanton and D. Marculescu, "Quantized deep neural networks for energy efficient hardware-based inference," 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018, pp. 1-8.
8. C.-D. Tsai, T.-Y. Chen, H.-W. Fu and T.-C. Huang. "TCBNN: Error-Correctable Ternary-Coded Binarized Neural Network," 2021 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS2021), virtual, June 7, 2021.
9. D. T. Brown, "Error detecting and error correcting binary codes for arithmetic operations," IRE Trans. Electron. Comput., vol. EC-9, pp. 333-337, Mar. 1960.
10. Nicolae, A., "PLU: The Piecewise Linear Unit Activation Function", arXiv e-prints, 2018.