Low-complexity neuron for fixed-point artificial neural networks with ReLU activation function in energy-constrained wireless applications

Wen-Long Chin1 | Qinyu Zhang1 | Tao Jiang2

1 Department of Engineering Science, National Cheng Kung University, Taiwan
(Email: wlchin@mail.ncku.edu.tw; wantyoubetter@gmail.com)
2 School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China

Correspondence
Tao Jiang, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China.
Email: tao.jiang@ieee.org

Abstract
This work introduces an efficient neuron design for fixed-point artificial neural networks with the rectified linear unit (ReLU) activation function for energy-constrained wireless applications. The fixed-point binary numbers and ReLU activation function are used in most application-specific integrated circuit designs and artificial neural networks (ANN), respectively. It is well known that, owing to involved computation intensive tasks, the computational burden of ANNs is ultra heavy. Consequently, many practitioners and researchers are discovering the ways to reduce implementation complexity of ANNs, particularly for battery-powered wireless applications. For this, a low-complexity neuron to predict the sign bit of the input of the non-linear activation function, ReLU, by employing the saturation characteristics of the activation function is proposed. According to our simulation results based on random data, computation overhead of a neuron using the proposed technique can be saved by a ratio of 29.6% compared to the conventional neuron using a word length of 8 bits without apparently increasing the prediction error. A comparison of the proposed algorithm with the popular 16-bit fixed-point format of the convolutional network, AlexNet, indicates that the computation can be saved by 48.58% as well.

1 | INTRODUCTION

Artificial intelligence (AI) techniques are of paramount importance for the future intelligent wireless communications that can extract essential radio information [1]. The work [2] proposes a deep neural network (DNN) for the outer receiver of underwater acoustic orthogonal frequency-division multiplexing (OFDM). The artificial neural network (ANN) is also a promising solution for cognitive radios (CRs) [3]. The spectrum sensing for CRs [4] is studied by the works [5, 6] using an ANN-based approach. Moreover, ANNs have been adopted in wireless sensor networks (WSNs) [7–9]. Supervised learning methods for localization and object targeting have been extensively applied in WSNs, such as in [7]. The intelligent machine learning (ML)-based medium access control (MAC) scheduling in WSNs is studied in [8]. The intrusion detection in WSNs is studied using the ML theory in [9]. Moreover, ANNs are also applicable for the equalization in physical layers [10–12].

The work [13] studies the indoor localization using the gated-convolutional neural networks (GCNNs).

The neural network is a network consisting of connected neurons [14]. Hence, the ANNs are structured and organized in layers, leading to a network consisting of hundreds or thousands of artificial neurons, which are connected with adjustable coefficients [15]. A typical neuron has weighted inputs, bias, activation function, and one output. Weights and bias are optimized according to some learning rule during training by minimizing the error between the ANN and labeled outputs. A large training set is usually required in a training stage. The weighted sum of the inputs plus the bias constitutes the activity strength of the neuron, which is then passed through the activation function to produce the output. The activation function captures non-linear relationship between its input and output, and converts its input into a more useful output.

In order to design a hardware accelerator for convolutional neural networks (CNNs) that is able to reduce energy...
consumption, the current common practice is to tradeoff the computation accuracy for energy reduction. The benefits of reduced accuracy include reduced data storage space and computational effort. The works [16, 17] optimize the accuracy of the kernel weights, while the works [18, 19] retrain the kernel weights and activations. Compared to the floating-point AlexNet, the performance losses of [16–19] in terms of the prediction accuracy are 19.2%, 3.7%, 11%, and 5.2%, respectively. Moreover, they all require retraining, which is a complicated process.

In addition to the computation burden, another issue for the accelerators of ANNs is the requirement of high memory bandwidth. Three categories are identified by the work [20] to reduce the data movement to/from the memory, including the convolution reuse, fmap reuse, and filter reuse. The weight stationary and output stationary data flow were adopted by the works [21] and [22], respectively.

We aim to propose an efficient algorithm without needing any retraining to reduce the involved computational load, which is critical to the wide deployment of fixed-point ANN systems for battery-powered wireless applications. For example, the famous CNN, AlexNet, requires 720 millions multiply-and-accumulate operations. Owing to the incurred complexity, fixed-point hardware accelerators [22–24] are efficient realizations for the neural networks. By contrast, solutions based on CPUs and GPUs have a high power consumption, which are not suitable for some application fields with low power supply [25, 26]. The works [22, 23] propose the 16-bit fixed-point number format to implement the accelerator of AlexNet. The work [24] studies the non-linear quantization for the AlexNet. The work [27] studies the stochastic computing, which uses the probability of 1s in a number to represent a random binary data, to replace conventional arithmetic operations like multiplication in LeNet-5 with simple logic operations. A new stochastic multiplier is proposed in [28] to reduce the energy consumption incurred from the MNIST handwritten digital image. The work [29] provides the theoretical foundations to optimally shape the activation function in DNNs.

Motivated by the percentage of being zero for the output of each neuron in AlexNet, which is about 65.8%, we propose a novel neuron design to predict the sign bit of the input of the non-linear activation function, rectified linear unit (ReLU). The fixed-point binary numbers and ReLU activation function are used in most application-specific integrated circuit (ASIC) designs and ANNs, respectively. To save the computational load spent in those zero outputs, more specifically, based on the behaviour of the ReLU activation function, we obtain a partial result based on only a few bits of inputs and weights of a neuron, which is then used to predict the activation output. If the partial result is smaller than a predetermined threshold, the output of a neuron is simply set to zero and the remaining results need not be calculated. Otherwise, the remaining result is obtained and pluses with the partial result such that the exact neuron output is the same as that of the traditional one without adopting the proposed technique. In the worst case, only one additional comparator is required in the proposed approach; in most cases, a large amount of arithmetic operations can be saved. The design parameters are determined by an optimization formula. To take advantage of the proposed method, we also devise a circuit architecture for the proposed neuron.

Since a neuron is the fundamental element of neural networks, the proposed method can be adopted into many state-of-the-art neural networks. In short, the major contributions of this work are outlined as follows. (1) We propose a new efficient artificial neuron design without any retraining, which lies at the heart of battery-powered neural networks for wireless applications. The design parameters are optimized as well. (2) An embodiment of the architecture of the proposed design is demonstrated.

The rest of this article can be outlined as follows. Section 2 illustrates the proposed efficient artificial neuron algorithm and its circuit architecture, while Section 3 designs the parameters of proposed algorithm and evaluates its performance. Finally, Section 4 draws the conclusions of this paper.

2 | ARTIFICIAL NEURON

2.1 | Background

DNNs with more than one hidden layer are gaining worldwide attention in recent years, because they can deliver excellent accuracy in many AI domains [15]. DNNs, such as CNNs, are capable of learning high-level and abstract features from a hyperspectral image. The CNNs were inspired by the organization of the animal visual cortex, where only localized neurons are connected to the output in a convolutional layer, and the same set of weights is repeatedly used. Owing to the wide deployment of DNNs, superior performances are gained in many areas. However, DNNs come at the burden of ultra high computational complexity and storage requirement.

Although there is still a long way to go in understanding the brain, it is generally believed that neurons are the basic functional units of the brain. An artificial neuron is presented in Figure 1, where $Y$, $X_i$, and $W_i$ imitate the output axon, input axon, and synapse of a neuron, respectively, and $I$ denotes the number of weights and inputs. Aligning artificial neuron to a physiological neuron, $Y$, $X_i$, and $W_i$ are also referred to as the...
output activation, input activation, and weight, respectively. The weight coefficient that represents the strength of connection indicates the synapse load taking a positive value for excitation and a negative one for inhibition. Using bias, $B$, is just a good practice making it easier for the ANN to work efficiently. Biases in ANNs increase the capacity to solve versatile problems. A bias unit stores the value of $+1$ and is not connected to any previous layer. In this sense, it does not represent a true activity.

2.2 | Proposed algorithm

The fundamental and most computation intensive task of ANN is the matrix and/or vector multiplications. For example, the filtering in CNN can be regarded as the dot product, and the fully 2D convolutions in CNN can be regarded as the dot product, and the fully connected network can be treated as the matrix-vector product. Hence, the major computation of an artificial neuron, called the primitive in this article, is that shown in Figure 1, and it is written as

$$Y = \sum_{i=0}^{N-1} W_i X_i + B,$$

where $Y, X_i, W_i,$ and $B$ are $N$-bit signed words for a fixed-point ANN. Since the complexity of an adder is much lower than that of a multiplier, we emphasize the complexity of multiplications here. To obtain $Y$, $I$ multiplications with $N$-bit×$N$-bit are required in the original ANN primitive. Moreover, it can be observed that the outputs of many activation functions saturate, for example, the ReLU activation function shown in Figure 2. The output of the ReLU has its minimum value of zero when its input is lower than zero, which motivates us to predict the sign bit of the primitive output $Y$ using the most significant bits (MSBs) of its input arguments.

We consider the 2’s complement fixed-point realization of neural networks. Also, $Y, X_i, W_i,$ and $B$ are $N$-bit signed words. Let $X_i = \{X_{i,k}, X_{i,N-k}\}$, where $X_{i,k}$ and $X_{i,N-k}$ denote the MSB $k$ bits and remaining $(N-k)$ bits of $X_i$, respectively, $k = 1, 2, ..., N$, and $\{C, D\}$ denotes the concatenation operation of signals $C$ and $D$. Similarly, the weight $W_i = \{W_{i,k}, W_{i,N-k}\}$ and bias $B = \{B_k, B_{N-k}\}$ can also be partitioned into two parts with MSB $k$ bits and remaining $(N-k)$ bits. Knowing the binary representation of a decimal number, it can be proved (see Appendix) that the output $Y = Y_1 + Y_2$ can thus be separated into two parts, $Y_1$ and $Y_2$, which are, respectively,

$$Y_1 = \left(\sum_{i=0}^{N-1} W_{i,k}X_{i,k}\right) \times 2^{2(N-k)} + B_k \times 2^{N-k}$$  \tag{2}

and

$$Y_2 = \left[\sum_{i=0}^{N-1} (W_{i,k}X_{i,N-k} + W_{i,N-k}X_{i,k})\right] \times 2^{N-k}$$

$$+ \left(\sum_{i=0}^{N-1} W_{i,N-k}X_{i,N-k}\right) + B_{N-k}$$  \tag{3}

where $\times$ denotes the multiplication operation. The partial result, $Y_1$, is the accumulation-then-shift of the product of the MSBs of $W_i$ and $X_i$, that is, $W_{i,k}X_{i,k}Y_i$ and the addition of $B_k$. Likewise, the remaining partial result, $Y_2$, is the accumulation-then-shift of the products, $W_{i,k}X_{i,N-k}$ and $W_{i,N-k}X_{i,k}$, and $W_{i,N-k}X_{i,N-k}$, and the addition of $B_{N-k}$. To reduce the complexity when adopting the ReLU activation function, if the sign bit of $Y$ can be detected, the remaining arithmetic calculations (for $Y_2$) can be omitted and saved. Therefore, we propose to determine the sign bit of $Y$ based on $Y_1$ calculated from the MSBs of $X_i, W_i$, and $B$, that can approximate the original $Y$.

The proposed low-complexity primitive is summarized in Table 1, where $\eta$ is the predetermined decision threshold. Notably, compared to the original primitive, in the worst case, only one additional comparator is needed in the proposed primitive. In this case, the derived $Y$ of the proposed primitive is the same as that using the conventional ANN primitive; hence, there exists no prediction error. However, if the condition that $Y_1 < \eta$ satisfies, only $Y_1$ is calculated while $Y_2$ is not needed. Hence, the multiplication complexity, defined as the number of one-bit additions, reduces to $I$ multiplications with $k$-bit×$k$-bit operands, which appears to be approximately a ratio $k^2/N$ to that of the original ANN primitive.

Even the proposed algorithm can reduce the computational complexity incurred in a neuron, its implementation is not very

| TABLE 1 | Algorithm: Proposed primitive |
|----------|-------------------------------|
| input: $W_i, X_i,$ and $B$ |
| 1 Calculate $Y_1$; |
| 2 if $Y_1 < \eta$ |
| 3 $Z = 0$; |
| 4 end |
| 5 else |
| 6 Calculate $Y_2$; |
| 7 Calculate $Y = Y_1 + Y_2$; |
| 8 Calculate $Z = F(Y)$ using the ReLU; |
| 9 end |
| output: $Z$ |
straightforward. Even worse, directly implementing the proposed algorithm on the CPU or GPU backend might lower the operating speed owing to the involved if statement. Moreover, the operation based on an arbitrary word length is also not suitable for commercial CPUs or GPUs. Therefore, we propose a customized hardware, as that shown in Figure 3, to fully take advantage of the proposed algorithm. The customized hardware can easily operate on arbitrary word lengths. Additionally, the tasks for obtaining $Y_1$ and $Y_2$ can be pipelined using the circuit design technique so that the calculation of $Y_2$ can be saved if necessary.

The overall architecture of the proposed primitive is designed and presented on the left-hand side in Figure 3, where $F(\cdot)$ denotes the activation function and ACC denotes the accumulator circuit with enable control pin, EN, used to accumulate the product of two input operands, that is, the MSB $k$ bits or remaining $(N-k)$ bits of $W_i$ and $X_i$, for $i = 0, 1, \ldots, I-1$, $<<$ and $<$ denote the left-shift and comparison operators, respectively, and $\oplus$ denotes an adder. Notably, the architecture can be verified by the proposed algorithm in Table 1 and the 2’s complement fixed-point representations of $Y_1$ and $Y_2$. The shift left operation necessitates no extra hardware resources. There are one ACC and three ACCs for obtaining $Y_1$ and $Y_2$, respectively. Notably, the ACC circuit is disabled when its EN pin is false. The calculation of $Y_1$ is always enabled and its EN pin of ACC is thus tied to logic one (or the power supply voltage). The EN pins of three ACCs of $Y_2$ is connected to the inverse of the control signal $Y_2$ _disable_, which is true when $Y_1 < \eta$. In this case, the computational load of $Y_2$ can be totally eliminated. Finally, via a multiplexer, the output $Z$ selects zero when $Y_2$ _disable_ is true; otherwise, it selects the output of the activation function whose input is $Y$.

The architecture of ACC with enable control is shown on the right-hand side in Figure 3, where $\odot$ denotes a multiplier. The register $R$ has an enable control pin E. Within which, $D$ denotes the D-type flip-flop. When the registers are disabled, the operands of multipliers do not change, and hence, all computation loads involved in the ACC can be saved. Otherwise, new inputs are selected to calculate corresponding new result.

One can adopt another equivalent design skill, clock gating, to skip the calculation of $Y_2$. To further reduce the computational complexity, the serial mode that inputs one bit of the arguments at a time might even terminate the calculation of $Y_1$ using a smaller number of bits than $k$ determined by the proposed algorithm. Consequently, both the calculations involved in $Y_1$ and $Y_2$ might be reduced depending on the actual values of input arguments.

**TABLE 2** Algorithm: Threshold search

| input: $W_i$, $X_i$, and $B$ |
|---|
| 1 $\eta = 0$; |
| 2 for each convolution |
| 1 Calculate $Y_1$; |
| 6 Calculate $Y_2$; |
| 7 Calculate $Y = Y_1 + Y_2$; |
| 2 if $Y_1 < \eta$ and $Y > 0$ |
| 3 $\eta = Y_1$; |
| 4 end |
| 9 end |
| output: $\eta$ |

The architecture of ACC with enable control is shown on the right-hand side in Figure 3, where $\odot$ denotes a multiplier. The register $R$ has an enable control pin E. Within which, $D$ denotes the D-type flip-flop. When the registers are disabled, the operands of multipliers do not change, and hence, all computation loads involved in the ACC can be saved. Otherwise, new inputs are selected to calculate corresponding new result.
3 | ALGORITHM PARAMETER DESIGN AND PERFORMANCE EVALUATION

3.1 | Performance evaluation on random data

First, to evaluate the feasibility of the proposed ANN algorithm, Monte Carlo simulations were conducted using random data. We focus on the inference rather than training, since the inference is typically performed on edge devices with limited resources in contrast to the cloud. Two parameters, the number of bits, $k$, used to calculate $Y_1$ and the decision threshold $\eta$, need to be determined. Knowing weights and input data sets, we can use an optimization method to obtain them, as shown below.

$$
\max \quad \left(1 - \left(\frac{k}{N}\right)^2\right) P_r
$$
subject to $P_r \leq 0.01$

$k = 1, 2, \ldots, N$ and

$\eta = 0, -\delta, -2\delta, -3\delta, \ldots,$

where $\delta$ denotes the resolution of $\eta$, $P_r = P(Y_1 < \eta)$ denotes the probability of computation saving (CS), that is, the probability that $Y_1 < \eta$, $P(c)$ denotes the probability, and $P_r = P(Y_1 < \eta \cap Y \geq 0)$ denotes the detection error probability, where $\cap$ represents the intersection, that is, the probability of $Y_1 < \eta$ and $Y \geq 0$. The ratio of reduced complexity is $1 - (k/N)^2$. Hence, the objective function intends to maximize the product of the ratio of reduced complexity and the probability of CS. Notably, the sign bit of $Y$ is to be detected, so the maximum decision threshold $\eta$ is 0, while its minimum value is configured as $-0.2$ in this experiment, which shall depend on the dynamic range of $Y$. In this experiment, the choice of the resolution of $\eta$, that is, $\delta$, is $-0.0125$, which is ad hoc. A higher resolution may derive design parameters that can save more computation. However, the time spent for the parameter design also increases drastically owing to the grid search of the proposed algorithm for parameter design. Fortunately, the determination of the design parameters is a one-time job. Once they have been obtained, no computational burden is needed for the testing of ANN.

Obviously, the probability of CS becomes larger when $\eta$ increases. As $k$ increases, more computation is required for detecting the sign bit of $Y$. To enhance the CS capability, the objective function is maximized under the constraint that the detection error probability is less than 0.01, which guarantees that the output error does not occur too frequently. Notably, the detection error probability also depends on $k$ and $\eta$. The CS capability (or objective function) is maximized such that the probability of CS can be maximized while the number of used $k$ bits for $Y_1$ is minimized concurrently. Notably, the constraints of $P_r$ and the resolution of $\eta$ depend on the applicable scenarios and might need to be customized for specific applications.

Activation inputs and bias are randomly generated by uniformly distributed random variables in the interval $(-0.5, 0.5)$, while weights are randomly generated by Gaussian distributed random variables with zero mean and unity variance. We assume $I = 256$ and $N = 12$. The performance was averaged over 100,000 trials. The detection error probability generally depends on both $k$ and $\eta$. As shown in Figure 4, the detection error probability decreases when $k$ increases or $\eta$ decreases. As displayed in Figure 5, the CS probability mainly depends on $\eta$. To guarantee the detection error probability, there exists feasible combinations of $k$ and $\eta$. To enhance the CS capability, $k$ must decrease and $\eta$ should increase. Hence, among the feasible solutions, a pair of $(k, \eta)$ exhibiting the maximum CS capability, which is $(5, -0.0375)$, can be chosen. In this case, the multiplication complexity of the proposed design is 40.04% compared to that of the conventional one. For other $N$’s, that is, $N = 8, 16, 20, 24, 28, \text{and} 32$, the ratio becomes 29.6%, 43.9%, 46.1%, 49.4%, 49.5%, and 50.1%, respectively. It can be seen that the CS capability enhances when $N$ increases, which tends to increase the ratio of reduced complexity.

Then, we evaluate the effects of different sizes of ANN, including $I = 256, 1024$, and 4096, on the proposed method. Their ratios of multiplication complexity to that of the conventional ANN are 40.04%, 40.54%, and 41.27%, respectively, which slightly increases because the variance of output activations increases for a larger $I$. Consequently, the probability of CS, $P_r$, and hence, saved multiplication complexity, for a larger $I$ becomes larger accordingly.
### Parameters of the proposed method

| Conv 1 | Conv 2 | Conv 3 | Conv 4 | Conv 5 |
|--------|--------|--------|--------|--------|
| \((k, \eta)\) | \((10, -4.04)\) | \((9, -5.78)\) | \((9, -3.67)\) | \((9, -36.53)\) | \((9, -44)\) |

### Performance evaluation on AlexNet

Next, we evaluate the proposed method on the well-known CNN, AlexNet. The simulation software is MATLAB R2018b. Kernel weights and biases of AlexNet are also derived from those established in MATLAB database. The data sets are obtained from those used in [30]. In each simulation run, more than 3000 random images are tested.

To determine \(k\) and \(\eta\) effectively for the neurons, we propose to use the same \((k, \eta)\) in a layer. If \(k\) and \(\eta\) need to be optimized for each neuron, for practicality issue, the computation burden for determining them should be huge for a DNN. Moreover, too many different \(k\) and \(\eta\) will make the fixed-point design quite difficult. At the other end, if the same \((k, \eta)\) is used for all neurons, it cannot accommodate the statistical characteristics of the input activation and weight in different layers.

Even so, the global search of the design parameters proposed in the previous section is ideally optimum but it is still unaffordable for a large convolutional network, such as AlexNet. Therefore, we propose a heuristic algorithm shown in Table 2 to determine \(\eta\) for a given \(k\) in this experiment. The algorithm is intuitive that, when a prediction error occurs (indicated by \(Y_1 < \eta\) but \(Y > 0\)), \(\eta\) is lowered to \(Y_1\) such that the inference using the same data set for finding \(\eta\) is guaranteed not to be wrong, but the inference using another data set might still be wrong with a low probability because the determination of \(\eta\) is based on the worst-case design principle.

After knowing \(\eta\) for each candidate \(k\), the two-dimensional optimization problem in the global search reduces to an one-dimensional problem. Therefore, we can sequentially determine \(k\) for each layer, as the final result shown in Table 3. We used 500 images to find \(\eta\) of each candidate \(k\).

The proposed method is compared to the 16-bit fixed-point number format, which is a popular implementation for most CNN accelerators [23]. The adopted prediction accuracy is top-1 accuracy. That is, the recognized object by the AlexNet must be perfectly accurate. As a result, the multiplication complexity of the proposed design is 48.58% compared to that of the conventional 16-bit fixed-point quantization [23]. The performance of the AlexNet is commonly evaluated using the prediction accuracy. The accuracy performance of the proposed design is 62.5%, approximately 0.16% worse than that of the floating-point AlexNet, which is 62.6%. In addition to AlexNet, other operations for the accelerators of wireless applications using the fixed-point artificial neuron with ReLU activation function, such as convolutional and fully connected layers, can accommodate the proposed design to save the computational complexity.

Figure 6 demonstrates \(P_s\) of Conv 1 plotted against \(k\) and \(\eta\). Other convolutional layers have similar trend and they are omitted here. The figure clearly demonstrates the effects of \(\eta\) on \(P_s\). When \(\eta\) increases, \(P_s\) will increase accordingly. When \(k\) decreases, the required computation also decreases, but the prediction accuracy will decrease as well.

Figure 7 demonstrates the tradeoffs between prediction accuracy and reduction on computation of AlexNet using the proposed algorithm. The result of the right-most point is obtained using the parameters in Table 3; the result of the second right-most point is obtained by reducing one bit in Conv 5; the result of the third right-most point is obtained by reducing one bit in Conv 5 and Conv 4, and so on. Finally, the result of the left-most point is obtained by reducing one bit in Conv 5, 4, … , 1. The figure clearly shows that the prediction accuracy can be traded off for reduction on computation. When the prediction accuracy lowers, the ability for the reduction on computation enhances accordingly.
CONCLUSIONS AND FUTURE WORK

CNNs are popular techniques for future wireless applications, such as applications of AlexNet or other wireless algorithms. We propose a novel fixed-point neuron design, which is essential for the ASIC implementation of DNNs, for energy-constrained wireless applications. As a result, significant computational overhead can be reduced by the proposed method without apparent prediction error. Our future work will focus on implementing the proposed circuit and evaluating its power consumption and the ANN size that the proposed design can support. We will also evaluate other activation functions and different DNNs, such as VGGNet, GoogLeNet, and ResNet, as well.

ACKNOWLEDGEMENTS

The authors would like to thank the editor and reviewers for their helpful comments in improving the quality of this paper. This work was supported in part by the grant MOST 109-2221-E-006-181, Taiwan.

REFERENCES

1. Quek, T.Q.S., et al.: Editorial: Special issue on “artificial intelligence for future wireless communications and networking.” Elsevier Digit. Commun. and Netw. 1, (2019)
2. Zhang, J., et al.: Deep neural network-based underwater OFDM receiver. IET Commun. 13, 1998–2002 (2019)
3. He, A., et al.: A survey of artificial intelligence for cognitive radios. IEEE Trans. Vehicular Techn. 4, 1578–1592 (2010)
4. Chin, W.L., et al.: Iterative synchronization assisted detection of OFDM signals in cognitive radio systems. IEEE Trans. Vehicular Techn. 4, 1633–1644 (2014)
5. Mustafa, R., Jaglan, R.R., Agrawal, S.: Decision-fusion-based reliable CSS scheme in CR networks. IET Commun. 7, 947–953 (2019)
6. Zhu, X.-L., et al.: Channel sensing algorithm based on neural networks for cognitive wireless mesh networks. In: Proceedings of IEEE SMC'15, pp. 1–4, Dalian, China (2008)
7. Lu, C.-H., Fu, L.-C.: Robust location-aware activity recognition using wireless sensor network in an attentive home. IEEE Trans. Autom. Sci. Eng. 4, 598–609 (2009)
8. Kim, M.H., Park, M.-G.: Bayesian statistical modeling of system energy saving effectiveness for MAC protocols of wireless sensor networks. In: Proceedings of SNPD’09, pp. 233–245, Heidelberg, Germany (2009)
9. Kaplantzis, S., et al.: Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In: Proceedings of 3rd ISNNIP’07, pp. 335–340, Melbourne, Australia (2007)
10. Zhao, H.Q., et al.: Complex-valued pipelined decision feedback recurrent neural network for non-linear channel equalisation. IET Commun. 9, 1082–1096 (2012)
11. Zhao, H.Q., et al.: Equalisation of non-linear time-varying channels using a pipelined decision feedback recurrent neural network filter in wireless communication systems. IET Commun. 3, 381–395 (2011)
12. Wang, T., et al.: Deep learning for wireless physical layer: Opportunities and challenges. China Commun. 11, 92–111 (2017)
13. Chin, W.L., et al.: Intelligent indoor positioning based on artificial neural networks. IEEE Network. 6, 164–170 (2020)
14. Met, X., et al.: A survey and measurement study of GPU DVFS on energy conservation. Elsevier Digit. Commun. Netw. 3, 89–100 (2017)
15. Sze, V., et al.: Efficient processing of deep neural networks: A tutorial and survey. Proc. IEEE 12, 2295–2329 (2017)
16. Courbariaux, M., Bengio, Y., David, J.-P.: BinaryConnect: Training deep neural networks with binary weights during propagations. In: Proceedings of NIPS’15, pp. 3123–3131 (2015)
17. Li, F., Liu, B.: Ternary weight networks. In: Proceedings of NIPS Workshop on EMDN (2016)
18. Rastegari, M., et al.: XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Proceedings of ECCV’16, pp. 525–542 (2016)
19. Cai, Z., et al.: Deep learning with low precision by half-wave Gaussian quantization. In: Proceedings of CVPR’17, pp. 5406–5414 (2017)
20. Chen, Y.-H., et al.: Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In: Proceedings of ISCA’16, pp. 367–379 (2016)
21. Gokhale, V., et al.: A 240 G-ops/s mobile coprocessor for deep neural networks. In: Proceedings of CVPR’14, pp. 682–687 (2014)
22. Du, Z., et al.: ShiDianNao: Shifting vision processing closer to the sensor. In: Proceedings of ISCA’15, pp. 92–104 (2015)
23. Chen, T., et al.: DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In: Proceedings of ASPLOS’14, pp. 269–284 (2014)
24. Moons, B., et al.: Energy-efficient ConvNets through approximate computing. arXiv:1603.06777 (2016)
25. Xie, W., et al.: An energy-efficient FPGA-based embedded system for CNN application. In: Proceedings of EDSSC’18 (2018)
26. Zhang, L., et al.: XNORCONV: CNNs accelerator implemented on FPGA using a hybrid CNNs structure and an inter-layer pipeline method. IET Image Process. 1, 105–113 (2020)
27. Li, J., et al.: Hardware-driven nonlinear activation for stochastic computing based deep convolutional neural networks. In: Proceedings of IJCNN’17, pp. 1230–1236 (2017)
28. Li, B., et al.: Quantized neural networks with new stochastic multipliers. In: Proceedings of ISQED’18, pp. 376–382 (2018)
29. Unser, M.: A representer theorem for deep neural networks. Proc. IEEE 12, 2295–2329 (2017)
30. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. 115, 211–252 (2015)

How to cite this article: Chin W-L, Zhang Q, Jiang T.
Low-complexity neuron for fixed-point artificial neural networks with ReLU activation function in energy-constrained wireless applications. IET Commun. 2021;15:917–923. https://doi.org/10.1049/ietcmu2.12129

APPENDIX

Knowing the decimal value of \( X_i = X_{i,k} \times 2^{N-k} + X_{i,N-k} \) and the same representations can be used for \( W_i \) and \( B_i \), after substituting them into (1), we have

\[
Y = \sum_{i=0}^{N-1} \left( W_i \times 2^{N-k} + W_{i,N-k} \right) \left( X_{i,k} \times 2^{N-k} + X_{i,N-k} \right)
+ \left( B_i \times 2^{N-k} + B_{i,N-k} \right)
= \left( \sum_{i=0}^{N-1} W_i \times X_{i,k} \right) \times 2^{(N-k)} + B_i \times 2^{N-k}
+ \left( \sum_{i=0}^{N-1} \left( W_i \times X_{i,N-k} + W_{i,N-k} \times X_{i,k} \right) \right) \times 2^{N-k}
+ \left( \sum_{i=0}^{N-1} W_{i,N-k} \times X_{i,N-k} \right) + B_{N-k}
= Y_1 + Y_2.
\]  

(A.1)

The proof follows.