Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 700~706
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp700-706

Enhancing the ternary neural networks with adaptive threshold quantization

Son Ngoc Truong
Faculty of Electrical and Electronics Engineering, HCMC University of Technology and Education, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received May 6, 2025
Revised Jul 11, 2025
Accepted Oct 14, 2025

Keywords:

Deep neural network
Image recognition
Speech recognition
Ternary neural network
Binary neural network

ABSTRACT

Ternary neural networks (TNNs) with weights constrained to –1, 0, and +1 offer an efficient deep learning solution for low-cost computing platforms such as embedded systems and edge computing devices. These weights are typically obtained by quantizing the real weight during the training process. In this work, we propose an adaptive threshold quantization method that dynamically adjusts the threshold based on the mean of weight distribution. Unlike fixed-threshold approaches, our method recalculates the quantization threshold at each training epoch according to the distribution of real valued synaptic weights. This adaptation significantly enhances both training speed and model accuracy. Experimental results on the MNIST dataset demonstrates a 2.5× reduction in training time compared to conventional methods, with a 2% improvement in recognition accuracy. On Google Speech Command dataset, the proposed method achieves an 8% improvement in recognition accuracy and a 50% reduction in training time, compared to fixed-threshold quantization. These results highlight the effectiveness of adaptive quantization in improving the efficiency of TNNs, making them well-suited for deployment on resource constrained edge devices.

Corresponding Author:

Son Ngoc Truong
Faculty of Electrical and Electronics Engineering, HCMC University of Technology and Education
No 1 Vo Van Ngan Street, Thu Duc Ward, Ho Chi Minh City, Vietnam
Email: sontn@hcmute.edu.vn

1. INTRODUCTION

Deep neural networks (DNN) have achieved remarkable success in human-like tasks such as speech recognition and image recognition over the past decades [1]-[7]. Increasing the number of layers and parameters enhances neural network accuracy, leading to the development of DNNs. However, state-of-the-art DNN architectures require substantial computational resources and are typically deployed on high-performance processor such as graphic processing units (GPUs) [8]. This is due to the large number of operations, including additions, multiplications, and activation functions, which demand significant processing power and storage. Consequently, deploying DNNs on low-cost computing platforms, such as embedded systems and edge devices, remains a challenge.

To address this issue, various optimization techniques have been proposed, including quantization, pruning, and distillation [9]-[17]. Among these, quantization is particularly well-suited for low-cost embedded systems, which are usually used for IoT end-nodes and edge devices. By reducing full-precision synaptic weights and activations to as few as two or even one bit, quantization minimizes computational complexity and memory requirements. Binary neural networks (BNNs), where both weights and activations are quantized to a single bit, have been introduced to significantly accelerate computations by replacing multiplications with simple logical XOR operations [13]-[17]. However, BNNs suffer from reduced accuracy

compared to full-precision networks. To bridge this accuracy gap, ternary neural networks (TNNs) have been proposed, where weights and activations are constrained to –1, 0, and +1 [18]-[21]. For image recognition task, TNNs achieve accuracy within 2% of full-precision networks, offering a promising trade-off between computational efficiency and model performance [21].

The conventional backpropagation algorithm with gradient descent cannot be directly applied to binary or TNNs, as gradients descent relies on small weight updates, which are not feasible with discrete binary or ternary weight values. Typically, synaptic weights are first updated with small values and then binarized using a sign function [15], enabling gradient-based training for such networks. In this work, we present a TNN for image recognition and speech recognition that can be deployed on low-cost embedded systems and edge devices. The synaptic weights are quantized to –1, 0, and +1, and an adaptive threshold quantization method is proposed to enhance recognition accuracy and accelerate training. The use of a low-cost embedded computing platform demonstrates the feasibility of deploying TNNs in resource-constrained environments, where computational efficiency and power consumption are critical factors. By leveraging ternary weight quantization and adaptive thresholding, our approach ensures a balance between model accuracy, training efficiency, and hardware suitability, making it well-suited for robotics applications requiring onboard neural network inference.

2. METHOD

DNNs with full-precision synaptic weights require substantial storage space and huge computational resources due to their reliance on 32-bit floating-point representations for both multiplications and additions. Each weight in fully-connected layer is typically represented with a high precision, and every forward and backward pass involves numerous floating-point operations, leading to significant memory consumption and computational overhead. As the depth of neural network increases, the demand for memory and processing power escalates exponentially, imposing severe limitations on hardware with limited resources. Furthermore, executing a large number of floating-point computations results in increased energy consumption and reduced processing speed, making full-precision DNNs impractical for deployment on power-constrained devices.

To address these challenges, TNNs have emerged as an optimized alternative, offering a balance between computational efficiency and model accuracy. In a TNN, synaptic weights are quantized to three discrete values, typically –1, 0, and +1, significantly reducing memory requirements and eliminating the need for high-precision floating-point multiplications. Instead, the computational workload is simplified to additions and sign-based operations, which are more efficient and hardware-friendly. Due to their low memory footprint and reduced computational complexity, TNNs can be effectively deployed on low-cost edge devices, where storage space, energy efficiency, and real-time processing capabilities are critical constraints.

Figure 1 presents a conceptual diagram of a ternary DNN, in which synaptic weights are constrained to three discrete values: –1, 0, and +1. In the convolutional layer, the kernel is a matrix of –1, 0, or +1 instead of full-precision weights. This quantization reduces the amount of storage required for model parameters and simplifies the computation of convolution operations. Similarly, in the fully-connected layer, the synaptic weights are also limited to –1, 0, or +1, ensuring that every layer of the network adheres to the ternary constraint. Ternary synaptic weights can be interpreted in terms of neuronal functionality, where the negative and the positive weights correspond to inhibitory and excitatory neuronal synapses, respectively, mirroring the way biological neurons regulate signal transmission [22], [23]. The synaptic weights are negative or positive for respectively representing the inhibitory or excitatory neuronal synapses [22], [23]. Additionally, synaptic weights assigned a value of zero represent unconnected synapses, effectively removing certain connections from the network. This mechanism is similar to the dropout technique used in DNNs, where randomly deactivating neurons during training helps prevent overfitting and enhance model generalization [24].

The TNN can be trained using the conventional backpropagation algorithm combined with a gradient descent-based optimization method. However, a fundamental challenge arises due to the nature of gradient descent: it updates synaptic weight using small real-valued increments, whereas ternary weight must be discretely switched among –1, 0, and +1. This discrepancy necessitates a specialized approach to weight updates to ensure effective training while maintaining ternary constraints. To address a similar challenge in BNN, Courbariaux et al. proposed a training methodology that restricts synaptic weights to –1 and +1 while still leveraging the backpropagation algorithm with gradient descent method [15]. In this approach, the network is trained using conventional weight update rule, where real-valued weight updates are computed in the backward pass, the synaptic weights are binarized using simple threshold function.

$$w_b = sign(w_r) \quad (1)$$

Where w_b represents a binary synaptic weight and w_r denotes a real-valued synaptic weight. In (1), a sign function with the zero-valued threshold is utilized to determine the corresponding binary weight values. Specifically, if the real-valued w_r is negative, the resulting binary synaptic weight w_b is assigned a valued of –1. Conversely, if w_r is positive, w_b is set to +1. This simple binarization method ensures that the weight values remain within the required constraints while still allowing the network to be trained conventional optimization technique.

The proposed method, as formulated in (1), is not only applicable to BNN, but can also be extended to facilitate the training of TNN. By incorporating an additional step, ternary weight values are derived from real-valued synaptic weights, enabling the model to represent three discrete values of –1, 0, and +1. This quantization process is formally expressed in (2).

$$w_t = \begin{cases} -1, if\, w_r <= -w_{threshold} \\ 0, if\, -w_{threshold} < w_r < w_{threshold} \\ +1, if\, w_r >= w_{threshold} \end{cases} \quad (2)$$

Here $w_{threshold}$ represents the threshold weight, w_r denotes the real-valued weight, and w_t is the ternary weight, which can take on one of three discrete values: of –1, 0, or +1.

It is important to note that there is no universally optimal method for selecting the threshold value. In practice, the threshold is typically chosen through empirical analysis to ensure that the training process converges efficiently with the fewest possible iterations while achieving the high accuracy. The optimal threshold often depends various factor, such as the dataset, model architecture, and the training condition. Additionally, as training progresses, the distribution of synaptic weights evolves across iterations. This phenomenon can be observed in Figure 2 which illustrates the variation in weight distributions during different training stages. In Figure 2, an epoch refers to a complete cycle in which all samples in the dataset are passed through the neural network during both the forward and backward propagation steps. The changes in weight distribution across epochs highlight the dynamic nature of the learning process in TNNs.

Figure 2(a) illustrates the distribution of real-valued synaptic weights after five epochs of the training process. The synaptic weights are distributed within the range of –1 to +1, with a high density of values concentrated around zero. This distribution pattern suggests that a significant proportion of synaptic weights remain close to zero during training, which may influence the quantization process and the overall performance of the network. Figure 2(b) provides a comparative analysis of synaptic weight distributions at two different stages of training: the 5th epoch and the 10th epoch, represented by black and red lines, respectively. These variations indicate that the real-valued synaptic weights undergo continuous adaptation throughout the training process, leading to dynamic shifts in their distribution. A crucial implication of this changing distribution is the impact on weight quantization. If the threshold value for quantization remains fixed throughout training, the mapping of real-valued weights to ternary values (–1, 0, +1) will yield inconsistent results at different training stages. As a result, this inconsistency can slow down the convergence of the network and negatively affect overall accuracy. Furthermore, selecting an excessively large threshold value increases the number of synaptic weights quantized to zero, effectively removing a larger portion of connections in the network. This excessive sparsity can significantly degrade the model's learning capability, leading to a substantial drop in recognition accuracy.

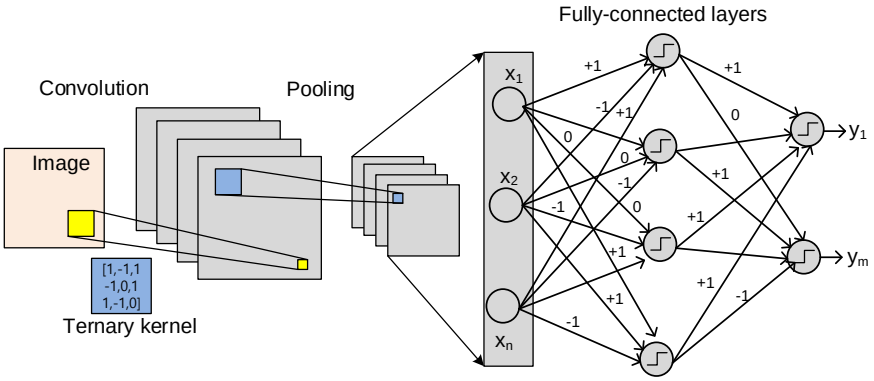

Figure 1. The concept of a TNN for image recognition

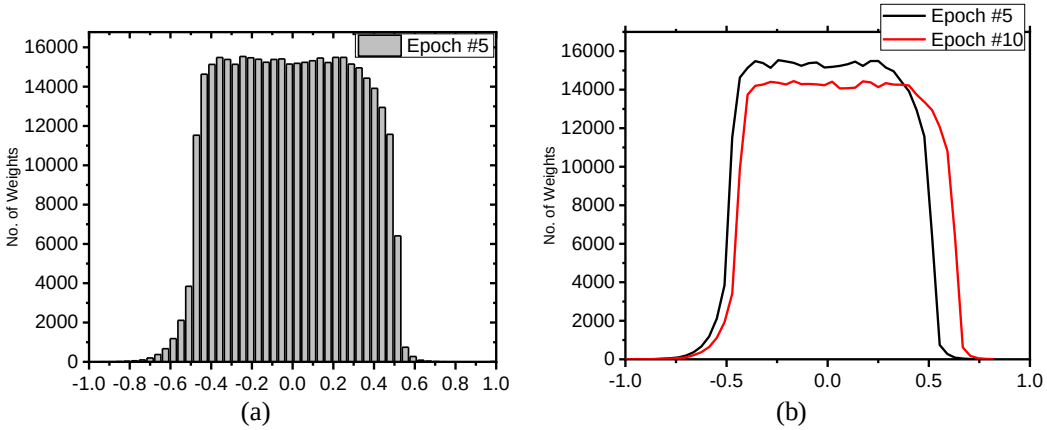

Figure 2. Variation in weight distributions during different training stages (a) The distribution of synaptic weights after the 5th epoch of the training process, (b) the distribution of synaptic weights after the 5th and the 10th epochs of the training process

In this work, we propose an adaptive thresholding method based on the Gaussian distribution to improve the quantization process in TNNs. Unlike fixed threshold approaches, which may lead to inconsistent weight distributions across training epochs, the proposed method dynamically updates the threshold epoch by epoch. This ensures that the proportions of negative synaptic weights (–1), zero-value synaptic weights (0), and positive synaptic weights (+1) remain approximately constant throughout the training process. The threshold value is determined based on the Gaussian distribution of the real-valued synaptic weights at each training epoch. By leveraging the statistical properties of the weight distribution, this method allows for an adaptive adjustment of the threshold, ensuring a more stable and balanced quantization process. A conceptual representation of this adaptive thresholding approach is illustrated in Figure 3, demonstrating how the Gaussian distribution guides the selection of threshold values for improved training convergence and model accuracy.

To determine the adaptive threshold values, the mean (μ) and standard deviation (σ) of the real-valued synaptic weights are first computed at each training epoch. Based on the properties of the Gaussian distribution, selecting threshold values at μ-0.44σ and μ+0.44σ ensures that the synaptic weights are quantized into three discrete categories with a balanced distribution. Specifically, this selection results in 33% of the synaptic weights being negative (–1), 34% being zero (0), and 33% being positive (+1). By maintaining this distribution, the proposed quantization method prevents excessive sparsity or imbalance in the weight representation, which could otherwise degrade the performance of the neural network. The quantization process follows the mathematical formulation given in (3).

$$w_t = \begin{cases} -1, if\, w_r <= \mu - 0.44\sigma \\ 0, if \mu - 0.44\sigma < w_r < \mu + 0.44\sigma \\ +1, if\, w_r >= \mu + 0.44\sigma \end{cases} \quad (3)$$

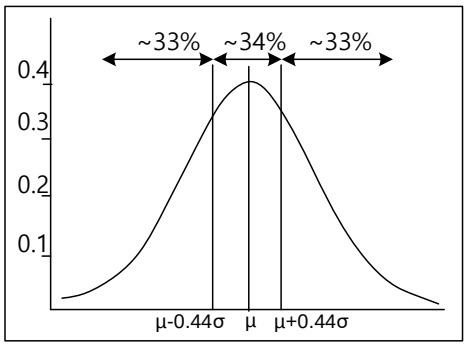

Figure 3. The Gaussian distribution of synaptic weights. The threshold is chosen to be μ-0.44σ and μ+0.44σ

Here μ and σ represent the mean and standard deviation, respectively, of real-valued synaptic weights at each training epoch. In (3), the threshold is dynamically changed epoch by epoch of the training process. Here the threshold is set to be (μ–0.44σ) and (μ+0.44σ). By applying this adaptive thresholding approach, the proportions of negative (–1), zero-value (0), and positive (+1) synaptic weights remain approximately constant in every epoch. This stability in weight distribution helps mitigate issues associated with fixed-threshold quantization, such as training instability and performance degradation. As a result, the proposed method not only improves the accuracy of the TNN but also accelerates the training process, enabling more efficient convergence.

3. RESULTS AND DISCUSSION

To highlight the advantages of TNNs over full-precision neural networks, we first compare their inference times as they are deployed on a resource-constrained edge device. A convolutional neural network (CNN) consisting of 64 convolutional kernels of size 3×3, followed by a max pooling layer and a fully connected layer with 512 hidden units and 10 output units, is implemented on a low-cost edge device, the Raspberry Pi 5, for handwritten character recognition using the MNIST dataset. The Raspberry Pi 5 is equipped with a 64-bit ARM processor running at 2.4 GHz, making it a viable platform for edge computing applications. Both the full-precision and TNNs are trained on a server. The pre-trained models are implemented on the Raspberry Pi 5 to evaluate inference performance. Experimental results indicate that the full-precision CNN requires 3.2 ms to classify a single character, whereas the ternary CNN achieves the same task in only 0.84 ms. These findings demonstrate that the TNN significantly outperforms the full-precision model on resource-constrained edge devices. The advantages of TNNs have also been discussed in previous studies [20], [21].

One of the key advantages of edge computing is the ability to perform both training and inference of DNNs directly on the edge device. In this study, we train the TNN using both the conventional fixed-threshold quantization method and the proposed adaptive threshold quantization approach on the MNIST dataset [25].

Figure 4 shows a comparison of the accuracy between the fixed-threshold quantization and the proposed adaptive threshold quantization over 100 training epochs. The accuracy of the fixed-threshold approach, represented by the black line in Figure 4, reaches 94% after 100 epochs. In contrast, the proposed adaptive threshold quantization achieves 97% accuracy in only 40 epochs. This demonstrate that the proposed method improves accuracy by 3% compared to the fixed-threshold approach. Moreover, training a TNN with the adaptive threshold quantization method is 2.5× faster than with the fixed threshold quantization. The ternary CNN is also evaluated on the Google Speech Commands dataset, which consists of 65,000 audio samples spanning 30 distinct words [26]. To ensure compatibility with resource-constrained edge device, a CNN is designed with 4 convolutional layers, a fully-connected layer with 1024 neurons, and a SoftMax layer with 30 output neurons. Mel-frequency cepstral coefficients (MFCCs) are used for feature extraction. The extracted coefficients are quantized by 8 bits.

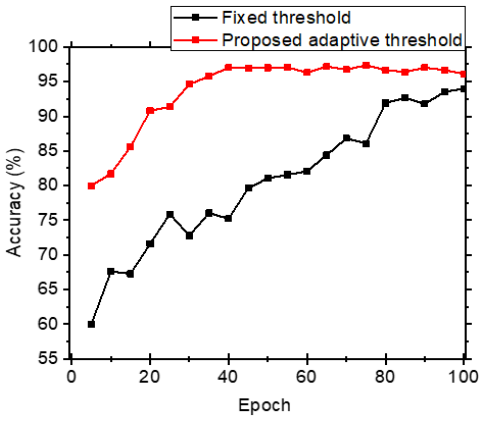

Figure 4. The comparison of the accuracy between the fixed threshold quantization and the proposed adaptive threshold quantization for 100 epochs of the training process

Figure 5 illustrate the accuracy of the ternary convolution neural network for speech command recognition. Using the proposed adaptive threshold quantization, the model achieves a recognition rate of 91% after 120 epochs. In contrast, with the fixed threshold quantization method, the recognition rate reaches 83% after 200 epochs, demonstrating an 8% improvement with the proposed approach. These results represent the first evaluation of TNNs with adaptive threshold quantization for object recognition for edge computing. The ternary CNN can be efficiently deployed on a resource-constrained edge device such as Raspberry Pi. The proposed adaptive threshold quantization method enhances both accuracy and training efficiency, making it a promising technique for low-cost, resource-constrained in edge computing.

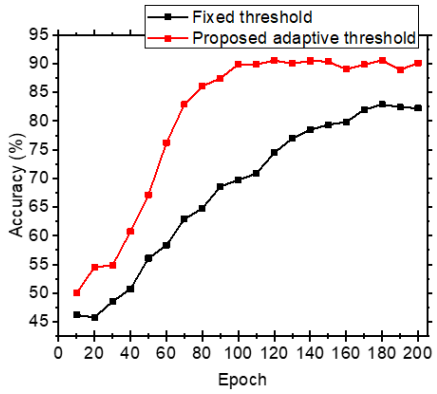

Figure 5. The comparison of the accuracy for speech recognition between the fixed threshold quantization and the proposed adaptive threshold quantization for 200 epochs of the training process

4. CONCLUSION

In this paper, we present a ternary CNN that constrains synaptic weights to –1, 0, and +1 for both image and speech recognition, using the proposed adaptive threshold quantization. The ternary CNN is deployed on low-cost and resource-constrained computers for edge computing. Training is performed using conventional backpropagation algorithm with gradient descent, alongside the proposed adaptive threshold quantization method, which dynamically adjust quantization thresholds based on the distribution of real-valued synaptic weights. For MNIST image recognition, training the TNN with adaptive threshold is 2.5× faster and achieve a 2% higher recognition rate compared to fixed threshold quantization. For speech recognition, the proposed method improves recognition rate by 8% and accelerates training by 1.6× relative to the fixed-threshold approach. These results demonstrate that the proposed TNN, equipped with adaptive threshold quantization, is an effective solution for deployment on resource-constrained edge computing systems.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Author state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceding Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, no. 2, pp. 1764–1772, 2014.

[4] P. B. Patil, "Multilayered network for LPC based speech recognition," *IEEE Transactions on Consumer Electronic*, vol. 44, no. 2, pp. 435–438, 1998.
[5] Y. Said, M. Barr, and H. E. Ahmed, "Design of a face recognition system based on convolutional neural network (CNN)," *Engineering, Technology & Applied Science Research*, vol. 10, no. 3, pp. 5608–5612, Jun. 2020, doi: 10.48084/etasr.3490.
[6] A. Hannun *et al*., "Deep Speech: Scaling up end-to-end speech recognition," *Computing Research Repository*, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.5567.
[7] M. Volodymyr *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015, [Online]. Available: http://www.nature.com/nature/journal/v518/n7540/abs/nature14236.html.
[8] H. Jang, A. Park, and K. Jung, "Neural network implementation using CUDA and OpenMP," in *2008 Digital Image Computing: Techniques and Applications*, 2008, pp. 155–161, doi: 10.1109/DICTA.2008.82.
[9] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: towards lossless cnns with low-precision weights," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.
[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," *Fiber*, vol. 56, no. 4, pp. 3–7, 2015.
[11] X. Dai, H. Yin, and N. K. Jha, "Grow and prune compact, fast, and accurate LSTMs," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 441–452, Mar. 2020, doi: 10.1109/TC.2019.2954495.
[12] J. Zhong, G. Ding, Y. Guo, J. Han, and B. Wang, "Where to prune: using LSTM to guide end-to-end pruning," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul. 2018, vol. 2018-July, pp. 3205–3211, doi: 10.24963/ijcai.2018/445.
[13] Y. Wang, J. Lin, and Z. Wang, "An energy-efficient architecture for binary weight convolutional neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 280–293, Feb. 2018, doi: 10.1109/TVLSI.2017.2767624.
[14] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, Jun. 2019, doi: 10.3390/electronics8060661.
[15] M. Courbariaux and Y. Bengio, "Unknown - ._Binarized Neural Networks_ Training neural networks with weights and activations constrained to 1 or−1_20180118194148.pdf -," *arXiv*, 2016.
[16] C. Baldassi, A. Braunstein, N. Brunel, and R. Zecchina, "Efficient supervised learning in networks with binary synapses," *Proceedings of the National Academy of Sciences*, vol. 104, no. 26, pp. 11079–11084, Jun. 2007, doi: 10.1073/pnas.0700324104.
[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9906 LNCS, 2016, pp. 525–542.
[18] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and −1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2014, pp. 1–6, doi: 10.1109/SiPS.2014.6986082.
[19] S. Yin *et al.*, "An energy-efficient reconfigurable processor for binary-and ternary-weight neural networks with flexible data bit width," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1120–1136, Apr. 2019, doi: 10.1109/JSSC.2018.2881913.
[20] S. N. Truong, "A low-cost artificial neural network model for Raspberry Pi," *Engineering, Technology & Applied Science Research*, vol. 10, no. 2, pp. 5466–5469, Apr. 2020, doi: 10.48084/etasr.3357.
[21] S. N. Truong, "A ternary neural network with compressed quantized weight matrix for low power embedded systems," *Engineering, Technology & Applied Science Research*, vol. 12, no. 2, pp. 8311–8315, Apr. 2022, doi: 10.48084/etasr.4758.
[22] L.~F.~Abbott and W.~G.~Regehr, "Synaptic computation," *Nature*, vol. 431, pp. 796–803, 2004.
[23] R. Zucker, "Short-term synaptic plasticity," *Annual Review of Neuroscience*, vol. 12, no. 1, pp. 13–31, Jan. 1989, doi: 10.1146/annurev.neuro.12.1.13.
[24] G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, and N. Srivastava, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014, [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.
[25] Deng Li, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[26] P. Warden, "Speech commands: a dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018, [Online]. Available: http://arxiv.org/abs/1804.03209.

BIOGRAPHIES OF AUTHOR

Son Ngoc Truong received his B.S. and M.S. degrees in Electronics Engineering from the Ho Chi Minh City University of Technology and Education, Vietnam, in 2006 and 2011, respectively. He completed his Ph.D. in Electronics Engineering at Kookmin University, Seoul, South Korea, in 2016. Dr. Truong was a postdoctoral researcher at Kookmin University from 2016 to 2017. He is currently an Associate Professor at Ho Chi Minh City University of Technology and Education, HCM City, Vietnam. His research interests include neuromorphic computing systems, brain-inspired circuits, hardware acceleration of deep neural networks, and related areas of electronic systems design. He can be contacted at email: sontn@hcmute.edu.vn.