LT-OAQ: Learnable Threshold based Outlier-Aware Quantization and its Energy-Efficient Accelerator for Low-Precision On-Chip Training

Qinkai Xu[1], Yijin Liu[2], Yuan Meng[1], Yang Chen[1], Yunlong Mao[3], Li Li[1], Yuxiang Fu[2]
[1]School of Electronic Science and Engineering, Nanjing University
[2]School of Integrated Circuits, Nanjing University
[3]State Key Laboratory for Novel Software Technology, Nanjing University
Email: {qinkaixu, yijinliu, yuanmeng, yangchen_nju}@smail.nju.edu.cn, {maoyl, lili, yuxiangfu}@nju.edu.cn

***Abstract*—Low-precision training has emerged as a powerful technique for reducing computational and storage costs in Deep Neural Network (DNN) model training, enabling on-chip training or fine-tuning on edge devices. However, existing low-precision training methods often require higher bit-widths to maintain accuracy as model sizes increase. In this paper, we introduce an outlier-aware quantization strategy for low-precision training. While traditional value-aware quantization methods require costly online distribution statistics operations on computational data, impeding the efficiency gains of low-precision training, our approach addresses this challenge through a novel Learnable Threshold based Outlier-Aware Quantization (LT-OAQ) training framework. This method concurrently updates outlier thresholds and model weights through gradient descent, eliminating the need for costly data-statistics operations. To efficiently support the LT-OAQ training framework, we designed a hardware accelerator based on the systolic array architecture. This accelerator introduces a processing element (PE) fusion mechanism that dynamically fuses adjacent PEs into clusters to support outlier computations, optimizing the mapping of outlier computation tasks, enabling mixed-precision training, and implementing online quantization. Our approach maintains model accuracy while significantly reducing computational complexity and storage resource requirements. Experimental results demonstrate that our design achieves a 2.9× speedup in performance and a 2.17× reduction in energy consumption compared to state-of-the-art low-precision accelerators.**

***Index Terms*—DNN, hardware accelerator, low-precision training, outlier-aware quantization**

I. Introduction

Deep Neural Networks (DNNs) have achieved remarkable success due to their exceptional performance in various domains, including computer vision [1] and natural language processing [2]. As DNNs grow in accuracy and capability, they present a significant challenge to existing computational platforms, placing greater demands on both storage capacity and computing performance [3]. Traditionally, training of these sophisticated networks has been conducted on cloud-based servers equipped with an array of high-performance GPUs, necessitating the transfer of private data from edge devices to the cloud. This approach raises significant challenges for data privacy and network bandwidth.

On-chip training has emerged as an effective solution to these challenges, enabling local device training or fine-tuning without uploading sensitive data to the cloud. Unlike inference tasks, training requires not only forward propagation but also backpropagation and weight updates, thus demanding bigger storage capacity and higher computational performance. To facilitate neural network training on edge devices with limited capabilities, model compression techniques such as pruning [4], quantization [5–8], and low-rank decomposition [9] are commonly employed to reduce performance requirements.

This work proposes a novel learnable threshold-based outlier-aware quantization (LT-OAQ) training method, based on the observed importance of outliers in models. LT-OAQ quantizes activations, weights, and gradients in low-precision data formats during neural network training. To avoid the costly online distribution statistics required by traditional outlier-aware quantization methods, we update outlier thresholds along with weights during training, adapting to dynamic input data and improving hardware efficiency. To fully exploit the potential of LT-OAQ, we have designed a hardware accelerator based on systolic arrays. This accelerator efficiently supports OAQ-Tensor format computations and mixed-precision training. It also performs effective online quantization of computational data, significantly reducing storage and computational overhead. As a result, the accelerator achieves substantial improvements in both performance and energy consumption. In summary, we make the following contributions in this paper:

1) We present a novel, learnable threshold based outlier-aware quantization (LT-OAQ) training framework that achieves an optimal balance between model accuracy and quantization precision, without relying on computationally expensive online data distribution statistics.
2) We introduce a hardware accelerator design tailored to support the LT-OAQ training framework, featuring a dynamic Processing Element (PE) fusion mechanism. This

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB2806802, in part by the Joint Funds of the National Nature Science Foundation of China under Grant U21B2032, in part by the National Nature Science Foundation of China under Grant 62104098, in part by the National Key Research and Development Program of China under Grant 2021YFB3600104, and in part by the Fundamental Research Funds for the Central Universities. (Corresponding authors: Yuxiang Fu; Li Li.)

978-3-9826741-0-0/DATE25/© 2025 EDAA

mechanism adaptively clusters adjacent PEs to efficiently handle outlier computations, optimize task mapping for various scenarios, facilitate mixed-precision training, and execute online quantization.

3) Through extensive experiment, we demonstrate that our LT-OAQ approach outperforms existing low-precision training methods and traditional value-aware techniques, achieving accuracy levels comparable to floating-point models while delivering substantial improvements, with up to 2.9× speedup in performance and up to 2.17× reduction in energy consumption compared to state-of-the-art low-precision accelerators.

II. Background

A. Low-Precision Training

High-precision data representation for gradients, typically using 32-bit floating-point numbers, is essential for maximizing network expressiveness and convergence speed during training. However, the computational demands of such precision make on-chip training impractical. Quantization offers a solution by reducing computational and storage requirements through the use of low-precision data formats for weights and activations [5]. To further minimize training overhead and bridge the performance gap between low- and full-precision networks, low-precision training methods are employed for fine-tuning [10] or training networks from scratch [8]. Equation (1) illustrates the application of quantization in the inference and training of neural networks. In this equation, s represents the scaling factor of quantization, and the clipping function $\lfloor x, min, max \rceil$ ensures that the data stay within the range $[min, max]$. The function $\prod()$ then maps the data to quantization levels $\mathcal{Q}_b$, where b stands for the precision of data.

$$\begin{aligned} \hat{w} &= \prod_{\mathcal{Q}_b} \lfloor \tfrac{w}{s}, min, max \rceil, \\ \hat{x} &= \prod_{\mathcal{Q}_b} \lfloor \tfrac{x}{s}, min, max \rceil, \\ \hat{out} &= \hat{w} * \hat{x}. \end{aligned} \tag{1}$$

In the backpropagation phase of neural network training, error loss values are quantized to low-precision data formats for error and gradient computations. When encountering non-differentiable operations, such as clipping, straight-through estimators (STE) [11] are typically employed to propagate errors directly.

Various quantization methodologies have been proposed to address the challenges of low-precision training. DoreFa-Net [7] utilizes fixed scaling parameters s to quantize weights, activations, and gradients across network layers. However, this approach lacks adaptability to highly variable input data, resulting in substantial accuracy degradation. PACT [6] enhances adaptability by implementing dynamic quantization by removing out-of-range data and continuous updates to the scaling parameter. Nevertheless, PACT's reliance on STE for computing gradients of the scaling parameter fails to account for the collective influence of data both within and beyond the clipping threshold. RV-Quant [12] takes a different approach. Instead of traditional linear quantization, it considers how outlier values contribute to the network's performance. This leads to a mixed-precision quantization method for low-precision training. NITI [8] introduces a comprehensive INT8 neural network training framework, employing per-layer scaling exponents to ensure adequate dynamic range. However, this method exhibits significant accuracy losses when applied to deeper, more intricate network architectures.

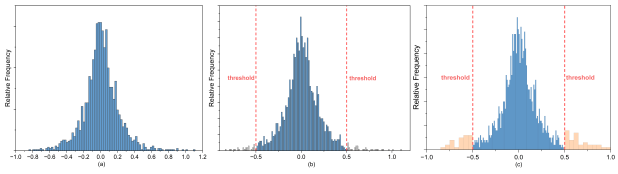

Fig. 1. Weight distributions of (a) full precision, (b) clipping-based quantization, and (c) outlier-aware quantization.

B. Neural Network Accelerator for Low-Precision Training

Recent research has presented various neural network accelerator designs, with a subset leveraging low-precision data formats for network training to improve computational performance and energy efficiency [13–17]. Notable among these is the HNPU [16], an adaptive deep learning neural network training accelerator that employs algorithm-hardware co-optimization for on-chip precision exploration and dynamic bit-width adjustment. However, this approach requires complex supplementary hardware for comparing low- and high-precision convolution outcomes, thereby escalating computational demands and power consumption during training. OLAccel [15] presents an alternative paradigm, focusing on outlier-aware quantization training methods. It introduces a hardware architecture capable of supporting variable precisions to manage outlier scenarios. Nevertheless, its reliance on discrete hardware units for outlier computations constrains resource utilization, thus limiting overall efficacy. Shao et al. [17] proposed a reconfigurable FPGA-based accelerator tailored for the NITI algorithm, facilitating deep neural network training using 8-bit integer data. While innovative, this accelerator's capability is confined to fixed precision computations, which imposes restrictions on its flexibility and potential for performance enhancement.

III. Learnable Threshold based Outlier-Aware-Quantization Training

The weight distribution during deep neural network training typically follows a Laplace distribution, characterized by a concentration of data near zero with a small subset of high-magnitude outliers, as Fig. 1 shows. Traditional linear quantization methods, in their pursuit of high quantization accuracy for the majority of near-zero data, often employ clipping techniques to a predetermined threshold, eliminating these outliers. However, this approach proves problematic for deep neural networks, particularly in classification tasks, where these outliers often exhibit winner-take-all characteristics and contribute more to the inference results [12]. Consequently, indiscriminate clipping of outliers can significantly compromise accuracy.

Outlier-aware quantization strategies address this challenge by conducting distribution statistics on the original data to determine an outlier threshold based on predefined proportions. This approach enables the application of distinct quantization precisions or methodologies to the majority of near-zero data and the outliers, achieving a balance between quantization range and precision.

In traditional outlier-aware inference or training strategies [12, 15], it is necessary to statistically analyze the distribution of the data involved in the computation to determine a threshold corresponding to the outlier ratio. Data exceeding this threshold are considered outliers, while those below are classified as normal. When completing inference tasks, the weight data exhibits a fixed distribution, allowing for offline statistical analysis. Unlike forward inference, the distribution of activations and weights changes during on-chip training, necessitating online statistical analysis. However, performing such distribution analysis in hardware introduces significant performance overhead. For instance, to maintain the computational efficiency of neural network accelerators, a substantial number of parallel numerical comparators and additional storage resources are required.

To enable efficient outlier-aware training, we propose a learnable threshold based outlier-aware training strategy (LT-OAQ). This strategy dynamically updates the outlier thresholds and model weights during on-chip training via gradient descent, thus avoiding the costly data re-statistics operations.

For forward inference, we have developed a hardware-optimized quantization algorithm, represented by (2). In this equation, α denotes a learnable outlier threshold, while the clipping function $\lfloor x, 1 \rceil$ normalizes the data to the interval $[-1, 1]$. Subsequently, it maps the data to quantization levels corresponding to precision b. For normal values, linear quantization is employed to achieve low-precision representation. Conversely, outliers are shifted proximal to zero and quantized using an adjusted scaling factor, simultaneously preserving high dynamic range and fine quantization granularity.

$$\hat{x} = \begin{cases} \alpha \prod_{\mathcal{Q}_b} \lfloor \frac{x}{\alpha}, 1 \rceil, & x < \alpha, \\ (x_{max} - \alpha) \prod_{\mathcal{Q}_b} \lfloor \frac{x-\alpha}{x_{max}-\alpha}, 1 \rceil, & x \geq \alpha. \end{cases} \tag{2}$$

To jointly optimize the model weights W and the outlier threshold α during training via gradient descent, we calculate the gradient of α using (3). Both outliers and normal values are optimized simultaneously through the clipping and mapping functions of the quantization process, aiming to achieve a balance between them.

$$\frac{\partial \hat{x}}{\partial \alpha} = \begin{cases} \prod_{\mathcal{Q}_b} \lfloor \frac{x}{\alpha}, 1 \rceil - \frac{x}{\alpha}, & x < \alpha, \\ \frac{x-\alpha}{x_{max}-\alpha} - \prod_{\mathcal{Q}_b} \lfloor \frac{x-\alpha}{x_{max}-\alpha}, 1 \rceil, & x \geq \alpha. \end{cases} \tag{3}$$

Unlike traditional outlier-aware quantization methods, (2) and (3) utilize only the maximum value of the raw data for quantization. As discussed in Section IV, this approach requires only a single comparator and additional storage for the maximum value, thereby avoiding the complexities and power consumption associated with more intricate hardware circuits.

The processing of outliers, which are data points with significantly larger amplitudes, presents a unique challenge in computational efficiency due to their typical requirement for higher precision representations. To address this issue while utilizing low-precision computational units, we separate outliers into two distinct components: tensor data expressed in OAQ-Tensor format and outlier thresholds, as formulated in (4).

$$\begin{aligned} Normal\ Tensor &= OAQ\text{-}Tensor(low\text{-}precision), \\ Outlier\ Tensor &= OAQ\text{-}Tensor(low\text{-}precision) \\ &\quad + threshold(high\text{-}precision). \end{aligned} \tag{4}$$

The OAQ-Tensor is amenable to lower bit-width representation, thereby facilitating computational efficiency. Conversely, the outlier thresholds are maintained in higher precision formats to preserve critical information. This division method enables our approach to achieve accuracy levels comparable to those of high-precision DNN models, while significantly reducing computational overhead.

Algorithm 1 Neural network training procedure for LT-OAQ training framework

Input: input activation X_{in}, weight W, outlier value threshold for activations and weights α_X, α_W.

Output: the output activations X_{out}.

Forward Propagation

Quantize the X_{in} with LT-OAQ quantization.
Quantize the W with LT-OAQ quantization.
Calculate the output X_{out}.

Backward Propagation

Calculate the loss L and error $\frac{\partial L}{\partial X_{out}}$
Calculate the gradient of activation and weight $\frac{\partial L}{\partial X_{in}}$, $\frac{\partial L}{\partial X_W}$
Calculate the gradient of outlier value threshold $\frac{\partial L}{\partial \alpha_X}$, $\frac{\partial L}{\partial \alpha_W}$

Parameters Update

Update weight $W = W - \eta \frac{\partial L}{\partial X_W}$
Update outlier value threshold
$\alpha_X = \alpha_X - \eta \frac{\partial L}{\partial \alpha_X}$, $\alpha_W = \alpha_W - \eta \frac{\partial L}{\partial \alpha_W}$

The detailed training process is described in Algorithm 1. In the forward pass, the LT-OAQ algorithm quantizes both activation values and weights. During backpropagation, the gradients of the outlier thresholds for each layer are computed along with the errors and gradients. Finally, gradient descent is used to jointly optimize the weights and outlier thresholds.

IV. Hardware Architecture

To facilitate efficient on-chip training, we have developed a hardware accelerator that supports matrix-multiplication, which accounts for the majority of the computational overhead in neural network operations, leveraging the LT-OAQ quantization strategy with a systolic array architecture [18]. Our design, illustrated in Fig. 2, comprises three primary components: Processing Elements(PEs), decoders, and outlier-aware quantizers. The accelerator processes data stored in the OAQ-Tensor format, which is initially decoded for computation. After

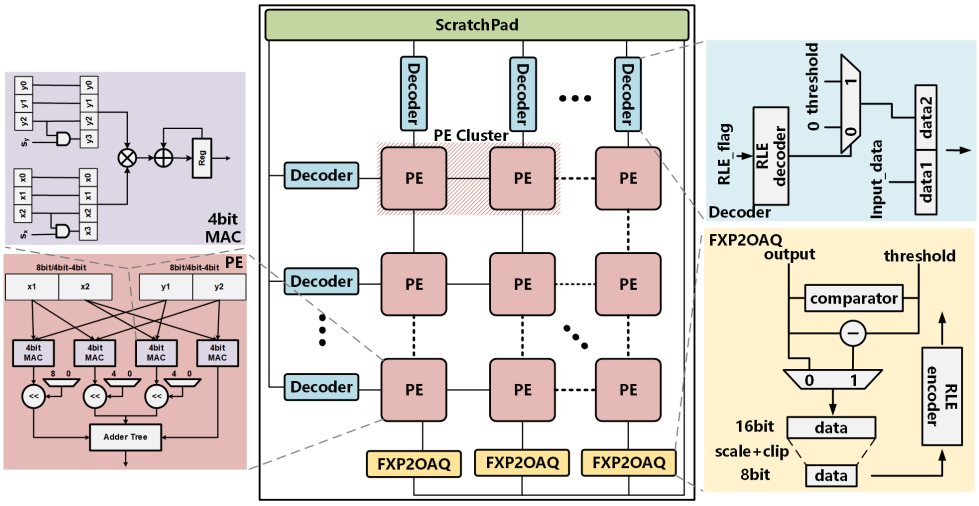

Fig. 2. Overall architecture of accelerator.

processing, the data are requantized to a low-precision OAQ-Tensor for storage optimization.

A. Processing Element

Outlier Support. Our architecture implements an approach to handle the outlier computations. As elucidated in (4), we split outliers into OAQ-Tensor and outlier threshold components, while normal values are represented solely by OAQ-Tensors. To accommodate the diverse computational scenarios arising from various combinations of normal values and outliers, we have innovated a dynamic fusion mechanism that combines two adjacent PEs into a PE Cluster.

Fig. 3 illustrates the mapping method handling computational tasks within the PE Cluster across different input configurations. We categorize these configurations into three primary scenarios: (1) both operands are normal values (data sets ① and ③), (2) one operand is a normal value and the other an outlier (data set ②), and (3) both operands are outliers (data set ④). In the first scenario (Fig. 3.a), where both operands are normal values, a single PE suffices for computation. To optimize hardware utilization, we distribute two sets of computational data (① and ③) to the two PEs within the PE Cluster. The second scenario (Fig. 3.b) involves a hybrid of normal and outlier values. Here, we strategically allocate the OAQ-Tensor and outlier threshold of the outlier to separate PEs within the PE Cluster, facilitating parallel computation with the normal value to minimize latency. For the rare but critical case where both operands are outliers (Fig. 3.c), we employ a temporal reuse strategy. This approach obviates the need for complex data scheduling logic by sequentially utilizing both PEs in the PE cluster to process the OAQ-Tensor and outlier threshold of input ④. The rarity of the third scenario minimizes such long-latency computations. Consequently, our accelerator achieves the substantial speedup demonstrated in Subsection V, underscoring the efficiency of our design in applications.

Mixed-Precision Support. LT-OAQ utilizes an outlier threshold represented at a higher precision than the OAQ-Tensor to maintain high accuracy. In our implementation, which employs

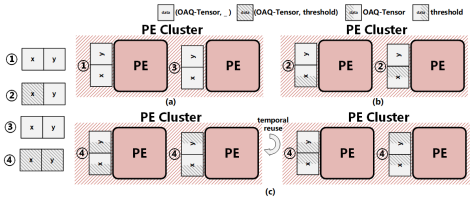

Fig. 3. Data scheduling with dynamic PE fusion for outlier support.

4-bit/8-bit quantization, the PE integrates four 4-bit MAC units that operate in parallel, adapting to the precision requirements of the input data. For 8-bit inputs, as shown in (5), the data are split into high and low 4-bit components, which are then processed separately before being recombined through shifters and adder trees. Conversely, 4-bit inputs are computed simultaneously across all four MAC units. This architecture effectively supports mixed-precision DNN computations, thereby enhancing overall network training accuracy.

$$\begin{aligned} A \times B &= (A_H \times 2^4 + A_L) \times (B_H \times 2^4 + B_L), \\ &= (A_H \times B_H \times 2^8) + (A_H \times B_H \times 2^4) \\ &\quad + (A_L \times B_H \times 2^4) + A_L \times B_L, \\ &= (A_H \times B_H << 8) + (A_H \times B_H << 4) \\ &\quad + (A_L \times B_H << 4) + A_L \times B_L. \end{aligned} \tag{5}$$

B. Outlier-Aware Quantizer

The accelerator employs an outlier-aware quantizer to maintain computational integrity and precision. During PE array computations, high-precision intermediate data representations are utilized to prevent overflow and preserve accuracy. The quantizer performs a critical comparison between the computed data and the outlier threshold, identifying outlier values. Upon detection, outliers undergo a threshold subtraction process, centering them around zero. This approach enables efficient low-precision representation while mitigating quantization-level waste. The final stage involves scale and clip operations to

truncate high bit-width data to the desired low-precision OAQ-Tensor.

Given the typically low frequency of outliers in the training process, the conventional method of allocating an additional bit to flag outliers introduces substantial overhead. This inefficiency is particularly pronounced when operating with low-precision data. To address this, our quantizer implements run-length encoding for outlier indices, which compresses consecutive normal value indices by retaining only the count of consecutive normal values while preserving outlier indices for the decoder to identify abnormal values, thereby optimizing hardware utilization.

C. Decoder

The on-chip computational data is stored in the OAQ-Tensor format. The decoder first processes the run-length encoded outlier indices to ascertain the type of each operand. When an outlier is identified, the decoder outputs both the OAQ-Tensor data and the corresponding outlier threshold. This triggers the PE computation unit to execute specialized calculations tailored for outlier values, ensuring accurate processing of these exceptional cases.

V. Experiment Results

In this section, we experimentally evaluate LT-OAQ in terms of model accuracy, area overhead, performance, and energy consumption.

A. Experiment Setup

We implemented our LT-OAQ training framework in PyTorch. The PE, decoder, and quantizer components of LT-OAQ, as described in Section IV, were implemented in Verilog RTL. These components were synthesized using Synopsys Design Compiler with TSMC 28nm technology to obtain area and static/dynamic power consumption estimates. To assess the energy consumption and actual execution cycles of the hardware accelerator, we developed a cycle-accurate simulator based on the DnnWeaver2 [19] framework, enabling end-to-end evaluation.

We evaluated LT-OAQ using a suite of six state-of-the-art vision-related DNN models, including VGG7, VGG16 [20], ResNet18, ResNet50 [21], InceptionV3 [22], and ViT-tiny [23]. These models were trained from scratch on the CIFAR-10 and CIFAR-100 datasets.

B. Accuracy Results

Quantization MSE. We conducted a comprehensive analysis of quantized mean squared error (MSE) using VGG16 on the CIFAR-10 dataset. Fig. 4 compares the quantized MSE values obtained through NITI's INT8 quantization methodology [8] and our proposed LT-OAQ approach. The results demonstrate that LT-OAQ consistently yields lower quantized MSE across DNN layers for both weights and activations. Even with 4-bit/8-bit mixed precision for normal/outlier values, LT-OAQ outperformed 8-bit NITI, particularly for signed weight data.

Model Accuracy. We benchmarked our method against NITI, RV-Quant approach based on outlier ratios, and floating-point models. To ensure an unbiased assessment of low-precision training efficacy, all networks were trained from scratch, without the utilization of pre-trained weights.

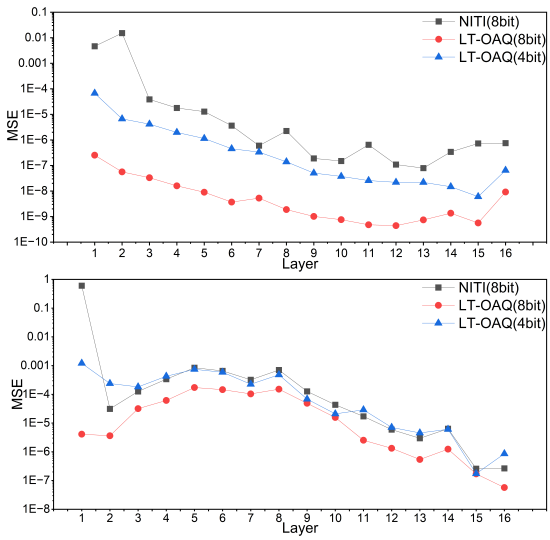

Fig. 4. Comparison of quantization mean square error (MSE) for (upper) weights and (lower) activations in VGG16.

Table I shows the comprehensive model accuracies. LT-OAQ training method consistently outperformed the extant low-precision training approach NITI across various model architectures and datasets. Notably, LT-OAQ achieved accuracies commensurate with, and in some instances surpassing, those of floating-point models. This can likely be attributed to the outlier-aware quantization's ability to minimize the loss of critical information, while simultaneously introducing a regularization effect that enhances the model's generalization performance. When compared with the computationally demanding RV-Quant training method, the LT-OAQ approach, which is more suitable for edge hardware deployment, demonstrated comparable accuracy. Further experimentation involving the reduction of normal value data bit-width to 4 bits for all layers, excluding the input and output layers, revealed that our quantization strategy yielded accuracy results approximating those of NITI with 8-bit.

TABLE I
Comparison of accuracy across various DNN models on Cifar-10 and Cifar-100 datasets.

Dataset	Model	float32	NITI	RV-Quant (costly)	LT-OAQ (8b/8b)	LT-OAQ (4b/8b)
Cifar-10	VGG7	91.68	90.53	91.75	92.52	91.81
	VGG16	92.44	87.45	90.11	91.41	90.50
	ResNet-18	93.05	92.91	93.55	93.26	92.33
	ResNet-50	93.2	92.63	93.43	92.88	87.35
	InceptionV3	95.37	90.07	92.8	93.47	86.71
	ViT-tiny*	87.19	82.87	85.86	85.81	80.1
Cifar-100	VGG7	70.72	63.57	67.29	69.15	65.39
	VGG16	66.38	58.16	59.29	63.85	61.01
	ResNet-18	75.21	67.59	71.58	73.24	69.07
	ResNet-50	76.89	66.04	73.21	72.83	64.24
	InceptionV3	78.6	67.11	73.09	73.74	63.18
	ViT-tiny*	63.18	54.71	61.78	60.23	52.87

*ViT-tiny didn't use any pre-trained weights.

C. Hardware Results

We implemented a systolic array computation circuit based on the architecture described in Section IV, with a 32×32 PE array. The synthesis report, detailed in Table II, reveals that the decoder and outlier-aware quantizer introduced to support LT-OAQ training methodology occupy a negligible area, thus incurring minimal hardware overhead. The hardware accelerator operates under conditions of 0.81V@500MHz, utilizing a 512KB on-chip buffer for data access.

TABLE II
ACCELERATOR PERFORMANCE SUMMARY.

Technology node	28nm
Supply Voltage	0.81V
Frequency	500MHz
Area	decoder:4419um^2
	FXP2OAQ:1661um^2
	PE:526450um^2
	total(logic):532530um^2
Buffer	512KB 4219461um^2

We scaled the hardware circuits of NITI [8] and OLAccel [15] to the equivalent area and technology node specifications for comparison. Simulations of DNN model training processes were conducted using a hardware simulator developed on the DnnWeaver2 framework. Concurrently, we employed CACTI [24] for simulations of latency and energy consumption in memory structures.

Performance. Fig. 5 presents normalized runtime across various hardware accelerators during training. LT-OAQ demonstrates superior latency performance, attributable to its efficient quantization strategy that enables model training with lower precision. This approach achieved a significant 2.9× speedup compared to NITI and a 1.91× improvement over OLAccel.

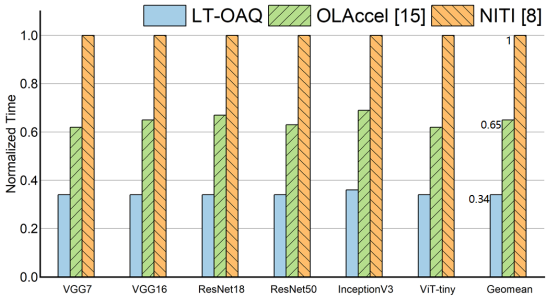

Fig. 5. Normalized execution time comparison of different accelerators.

Energy. Fig. 6 illustrates the normalized energy consumption during training, with a detailed breakdown of static energy and dynamic energy (DRAM, on-chip buffer, and core). Despite the increased buffer and DRAM access energy necessitated by online gradient computation and threshold updates, the LT-OAQ hardware accelerator achieves a 1.2× overall energy reduction compared to OLAccel, benefiting from simpler quantization, decoding, and control logic, which mitigates static power consumption. In comparison to NITI, LT-OAQ's lower precision computations result in reduced energy expenditure across all components, achieving a substantial 2.17× energy reduction.

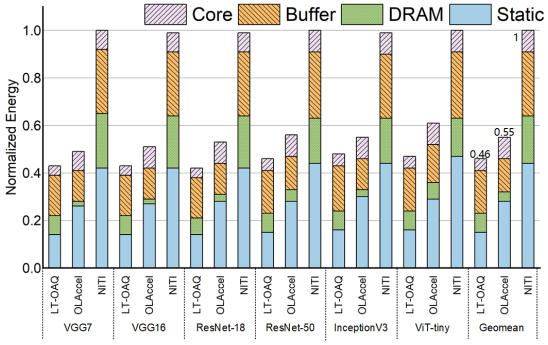

Fig. 6. Normalized energy consumption comparison of different accelerators.

VI. CONCLUSION

In this work, we introduce LT-OAQ, a novel low-precision training framework that achieves on-chip training accuracy comparable to floating-point models. The key innovation lies in recognizing the significance of outliers during the training process. To facilitate efficient on-chip training, LT-OAQ updates both weights and outlier thresholds simultaneously, eliminating the need for costly online data statistics operations. We have designed an on-chip training hardware accelerator based on a systolic array architecture that utilizes a dynamical PE fusion mechanism, optimizing computation scheduling for outliers and supporting mixed-precision training. Our design demonstrates up to 2.9× performance improvement and 2.17× energy reduction compared to state-of-the-art low-precision accelerators.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[2] K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[3] J. Lee and H.-J. Yoo, "An overview of energy-efficient hardware accelerators for on-device deep-neural-network training," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 1, pp. 115–128, 2021.

[4] S. Lym, E. Choukse, S. Zangeneh, W. Wen, S. Sanghavi, and M. Erez, "Prunetrain: fast neural network training by dynamic sparse model reconfiguration," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–13.

[5] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*. PMLR, 2015, pp. 1737–1746.

[6] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[7] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[8] M. Wang, S. Rasoulinezhad, P. H. Leong, and H. K.-H. So, "Niti: Training integer neural networks using integer-only arithmetic," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3249–3261, 2022.

[9] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6655–6659.

[10] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 941–22 954, 2022.

[11] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[12] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 580–595.

[13] D. Han, J. Lee, J. Lee, and H.-J. Yoo, "A low-power deep neural network online learning processor for real-time object tracking application," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1794–1804, 2018.

[14] J. Lee, J. Lee, D. Han, J. Lee, G. Park, and H.-J. Yoo, "An energy-efficient sparse deep-neural-network learning accelerator with fine-grained mixed precision of fp8–fp16," *IEEE Solid-State Circuits Letters*, vol. 2, no. 11, pp. 232–235, 2019.

[15] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 688–698.

[16] D. Han and H.-J. Yoo, "Hnpu-v1: An adaptive dnn training processor utilizing stochastic dynamic fixed-point and active bit-precision searching," in *On-Chip Training NPU-Algorithm, Architecture and SoC Design*. Springer, 2023, pp. 121–161.

[17] H. Shao, J. Lu, J. Lin, and Z. Wang, "An fpga-based reconfigurable accelerator for low-bit dnn training," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2021, pp. 254–259.

[18] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[19] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to fpgas," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] 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.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[23] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[24] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," 2009.