Threshold-Tuning Coordinate Attention for Binarized Neural Networks

Shaoqing Wu [1] and Hiroyuki Yamauchi [1,2,*]

[1] Intelligent Information System Engineering, Fukuoka Institute of Technology, Fukuoka, Japan
[2] Department of Computer Science and Engineering, Fukuoka Institute of Technology, Fukuoka, Japan
Email: bd24201@bene.fit.ac.jp (S.W.); yamauchi@fit.ac.jp (H.Y.)
*Corresponding author

***Abstract*—Binary Neural Networks (BNNs) are highly attractive for mobile and embedded vision due to their extremely low memory footprint and efficient bit-level convolutions. However, binarization often causes severe information loss and weakens the effectiveness of conventional attention modules designed for full-precision networks. We propose a BNN-oriented attention mechanism, Threshold-Tuning Coordinate Attention (TT-CA), which applies attention by adjusting the binarization decision boundary rather than performing fine-grained multiplicative reweighting. Built upon Coordinate Attention (CA), TT-CA generates a spatially aware threshold (τ) from coordinate-wise pooled features and applies a subtractive threshold to induce controllable sign flips, thereby recovering discriminative capability. To balance modulation capacity and deployment efficiency, we quantize the HardSigmoid gating outputs into K-bit discrete levels, enabling lightweight and hardware-friendly gating while avoiding the overly coarse behavior of 1-bit gates. We further explore CA-derived design variants under low-bit settings, including simplified normalization and lightweight bottlenecks, and integrate TT-CA into a compact ResNet14-Wide backbone. Experiments on CIFAR-100 and Tiny ImageNet show consistent accuracy gains over binarized baselines and standard attention plug-ins with modest overhead, reducing Top-1 error by 2.45% on Tiny ImageNet and 0.94% on CIFAR-100. Ablation studies on the threshold strength (δ) validate the effectiveness and robustness of threshold tuning for efficient BNNs.**

***Keywords*—residual neural networks, binary neural networks, coordinate attention, threshold tuning, edge AI, model compression**

I. Introduction

A. Background and Motivation

Convolutional Neural Networks (CNNs) have become the workhorse of modern vision systems [1]. High-performing models commonly stack repeated blocks with skip connections, as exemplified by ResNet, a deep CNN architecture introduced by He *et al*. [2] to address vanishing gradients and depth-related degradation through residual learning. In addition, many efficient CNN architectures rely on depthwise-separable kernels to reduce computation while representing weights and activations in 32-bit floating point (FP32). Despite their accuracy, deploying these networks on platforms with tight power and memory budgets remains challenging [3, 4]. Prior efforts in compression and quantization have helped, but the co-design of binarization and attention mechanisms has been comparatively underexplored, limiting simultaneous gains in accuracy and extreme computational and memory efficiency for binary CNNs [5].

Model quantization maps 32-bit Floating-Point (FP32) tensors to discrete low-bit representations, such as 8-bit Integer (INT8), sub-8-bit formats, and binary values, thereby reducing storage and computational costs with modest accuracy loss [6]. Work below 8-bit is active in research and is seeing early industrial adoption. At the 1-bit extreme binarized convolution can be executed with bitwise Exclusive-NOR (XNOR) [7] and population-count operations. Representative binary networks constrain weights and activations to {−1, +1} and introduce analytically derived or learned scaling factors to compensate for the information loss, thereby retaining much of the model's expressive capacity under such aggressive quantization.

Attention mechanisms have become a central tool in deep learning, enabling models to emphasize salient features within complex inputs. By learning data-dependent weighting, attention often improves both accuracy and computational efficiency.

The Squeeze-and-Excitation (SE) block [8], introduced by Hu and colleagues, adaptively reweights channels via global pooling (squeeze) followed by a lightweight gating (excitation), strengthening informative channels while suppressing noise.

Building on this idea, the Convolutional Block Attention Module (CBAM) proposed by Woo *et al.* [9] applies attention sequentially along the channel and spatial dimensions, yielding finer-grained feature emphasis. Although CBAM typically offers richer representations than channel-only SE, the additional spatial branch incurs extra computation and latency.

Manuscript received March 13, 2026; revised April 7, 2026; accepted May 19, 2026; published August 12, 2026.

Hou *et al.* [10] proposed Coordinate Attention (CA), which factorizes channel attention with directional (height/width) pooling to embed precise positional information into the channel descriptors. By generating axis aware gates and applying them multiplicatively, CA strengthens salient features while preserving location cues, delivering better accuracy–efficiency trade-offs especially in lightweight CNNs.

Conventional attention modules such as SE, CBAM, and CA were originally developed for full-precision models, and their fine-grained modulation ability is weakened in low-bit settings. Moreover, although their overhead is minor in large FP32 networks, it becomes relatively more significant in quantized models. While BNN-specific lightweight modules such as Binarized Ghost Module (BGM) [11] have also been proposed, this study focuses on improving conventional CNN attention modules.

We conducted experiments on CIFAR-100 [12] and Tiny ImageNet [13, 14]. As shown in Table I, in our binarized ResNet14-Wide, employing SE, CBAM, or CA actually degrades the performance of the model.

TABLE I. TOP-1 ERROR COMPARISON OF DIFFERENT ATTENTION MODULES IN BINARIZED RESNET14-WIDE

Model (ResNet14-Wide)	Tiny ImageNet Top1 Error	CIFAR-100 Top1 Error
Attention None	47.58 ± 0.28%	29.81 ± 0.19% (best)
SE	48.84 ± 0.31%	32.29 ± 0.32%
CBAM	48.40 ± 0.28%	31.87 ± 0.24%
CA	47.38 ± 0.3% (best)	31.52 ± 0.35%

Taking CIFAR-100 as an example, adding the SE module led to an average accuracy drop of 2.48%, while CBAM and CA reduced accuracy by 2.06% and 1.71% respectively, compared with the baseline without attention. On Tiny ImageNet, CA slightly improved accuracy by 0.20%, whereas SE and CBAM decreased accuracy by 1.26% and 0.82%, respectively.

For Tiny ImageNet, Wu *et al.* [15] reported that ResNet18 achieves a validation error rate of 43.5%, and Inception-ResNet achieves a validation error rate of 37.56%. In contrast, the VGG family models VGG16 and VGG19 obtain only 47.22% and 45.31% validation error, respectively, while also requiring a larger number of parameters. This is why we choose ResNet as the backbone for our study.

Overall, attention mechanisms originally designed for FP32 models tend to be counterproductive when applied to binarized networks. This motivates the need for an attention mechanism that remains effective in binary networks and can help recover the accuracy lost due to aggressive quantization. As shown in Table I, CA achieves better performance than SE and CBAM; therefore, we decide to develop our new attention module based on CA.

B. Proposed Threshold-Tuning Coordinate Attention (TT-CA)

We propose a controllable Threshold-Tuning Coordinate Attention (TT-CA) module (as shown in Table II) derived from Coordinate Attention (CA). Instead of multiplicative gating, we subtract a learnable threshold τ from the pre-binarization features, $X' = X - \tau$, with τ constrained by a strength parameter δ. This allows a small fraction of elements to flip sign when $|\tau| > |X|$, otherwise the sign is preserved. By enabling data-dependent, spatially aware sign changes, this threshold gating restores effective learnability to binarized networks.

TABLE II. COMPONENT-WISE COMPARISON BETWEEN CONVENTIONAL COORDINATE ATTENTION AND PROPOSED THRESHOLD-TUNING COORDINATE ATTENTION

Component	Conventional CA	Proposed TT-CA
Operation	Multiplication ($X \cdot A$)	Subtractive ($X - \tau$)
Function	ReLU	HardSwish
Mapping	Sigmoid	K-bit HardSigmoid
Output Range	Implicit (0–1)	Explicit K-bit
Hardware Efficiency	General	Tiny-AI oriented

To improve deployability and reduce memory traffic, we remove BatchNorm from the attention pathway and adopt HardSwish (bottleneck nonlinearity) and HardSigmoid (gating), both of which are piecewise-linear and thus quantization-friendly. We further quantize HardSigmoid to $k = 2$ bits using uniform levels $q \in \{0, \frac{1}{3}, \frac{2}{3}, 1\}$, mapping $g = 2q - 1$ provides symmetric values $\{-1, -\frac{1}{3}, \frac{1}{3}, 1\}$ better matched to binarized backbones. The final gate is used additively as a threshold $\tau = 1/2\ \delta(g_h + g_w)$ before the sign function, enabling targeted sign changes with strength controlled by δ.

As shown in Fig. 1(a), conventional multiplicative attention uses a [0,1]-valued attention map to rescale feature amplitudes. However, in binarized neural networks, such non-negative scaling cannot change the sign after binarization.

In binarized networks, multiplicative attention often has little effect—or can even hurt performance—because applying it before the sign operation with $A \in [0,1]$ gives $\mathrm{Sign}(A \odot X) = \mathrm{Sign}(X)$, leaving the sign decision unchanged and thus largely nullifying its impact after binarization.

Unlike the conventional attention mechanism shown in Fig. 1(a), which only rescales feature amplitudes through multiplication, the method proposed in Fig. 1(b) performs threshold tuning before binarization. By introducing a learnable threshold τ, TT-CA shifts the decision boundary of the sign function during training, so that $\mathrm{Sign}(X - \tau)$ can differ from $\mathrm{Sign}(X)$ for features near zero. In this way, TT-CA enables effective attention modulation in binarized neural networks.

The novelty and main contributions of this paper are summarized as follows:

(1) To improve deployability and reduce memory traffic, we remove BatchNorm from the attention pathway and adopt HardSwish (bottleneck nonlinearity) and HardSigmoid (gating), both of which are piecewise-linear and thus quantization-friendly.

(2) We further quantize HardSigmoid to $k = 2$ bits using uniform levels $q \in \{0, \frac{1}{3}, \frac{2}{3}, 1\}$; mapping $g = 2q - 1$ provides symmetric values $\{-1, -\frac{1}{3}, \frac{1}{3}, 1\}$ better matched to binarized backbones. The final gate is used additively as a threshold $\tau = \frac{1}{2}\delta(g_h + g_w)$ before the sign function, enabling targeted sign changes with strength controlled by δ.

(3) We propose subtracting a threshold τ from the features before binarization, allowing a small subset of activations to flip sign when $|\tau| > |X|$, thereby increasing the model's learnability.

(4) On Tiny ImageNet, the proposed TT-CA reduces the Top-1 error from 47.58% to 45.13%, i.e., by 2.45% (a 5.15% relative reduction). On CIFAR-100, the Top-1 error decreases from 29.81% to 28.87%, giving a 0.94% improvement (about 3.15% relative reduction).

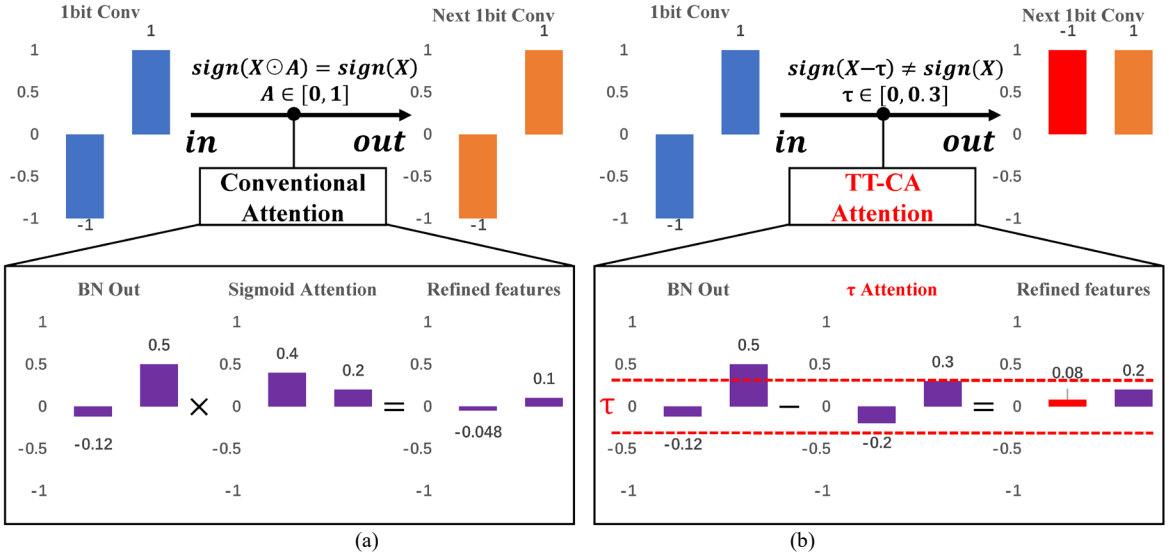

Fig. 1. Comparison between (a) conventional multiplicative attention and (b) the proposed TT-CA in binarized neural networks. While conventional multiplicative attention cannot change the sign after binarization, TT-CA changes the sign through threshold tuning.

C. Paper Organization

The remainder of this paper is organized as follows: Section II reviews binarization, the baseline model and its optimization, and attention mechanisms in binary networks. Section III introduces Coordinate Attention and presents the proposed TT-CA. Section IV describes the experimental setup, evaluates TT-CA applied to all BasicBlocks, and analyzes its effects at different network depths. Sections V and VI discuss the results and conclude the paper.

II. Related Work

A. Binarization

Traditional binarization schemes usually rely on $\mathrm{Sign}(W) \in [-1,1]$. In this case, all convolutional kernels are forced to have the same magnitude, which leads to more severe information loss. As a result, additional components such as Batch Normalization or subsequent layers are required to "recover" the scale, otherwise the performance degradation becomes more significant and convergence is harder to achieve [16–18]. The Straight-Through Estimator (STE) [19] alleviates this issue by enabling trainable gradients. Specifically, STE introduces a scaling factor α, computed for each output channel or kernel as $\alpha = \mathrm{mean}(|W|)$ over dimensions (1,2,3). In the forward pass, the weights are approximated as $\alpha \cdot \mathrm{Sign}(W)$, which preserves the energy and magnitude of the filters and significantly reduces the approximation error $\| W - \alpha \cdot \mathrm{Sign}(W) \|_2$, corresponding to the optimal α in the least-squares sense.

In this formulation, the forward path uses binarization with scaling, while the backward path bypasses the non-differentiable sign function by applying gradients to a clipped version of weight. This ensures that gradients are approximately one within the range [−1,1] and zero outside, resulting in more stable optimization. By assigning each output channel its own magnitude, the model can better approximate the distribution of real-valued convolution kernels, which is consistently shown to outperform the pure sign approach without α. Moreover, the scaled outputs are closer in magnitude to their real-valued counterparts (though more sparse and discretized), which allows better compatibility with Batch Normalization and activation functions. At inference, α can also be folded into subsequent BN layers, further reducing computational overhead.

In contrast, pure sign binarization introduces larger errors and weaker representational capacity. To compensate, models often require greater depth, wider architectures, or stronger data augmentation. This explains why, in our previous work under limited depth constraints, we resorted to wider networks to maintain performance.

B. Model Optimization

ResNet, is a deep CNN architecture built to counter vanishing gradients and depth-related degradation. By adding residual blocks with skip connections, information can bypass intermediate layers, which stabilizes training and enables substantially deeper, more expressive models.

Wide ResNet (WRN) [20] argues that reducing depth while widening layers can preserve or even improve accuracy and markedly shorten training time. In many settings, width offers better returns than depth, curbing overfitting and improving generalization. This perspective aligns with our ResNet14-wide design, which favors increased channel width over added depth—an approach well suited to deployment on resource-constrained hardware.

Building on the ResNet family, we construct a compact ResNet14-Wide as shown in Fig. 2, for mobile and embedded scenarios. The model achieves lightweighting by trimming depth and tuning per-stage channels while aiming to retain the baseline ResNet performance. ResNet14 removes the final residual block of ResNet18 and uses stage widths of 80, 160, and 320 channels, substantially cutting parameters and computation; by contrast, ResNet18 employs 64, 128, 256, and 512 channels. Although the network is shallower, widening the channels maintains representational capacity, allowing ResNet14-Wide to run efficiently on limited devices without materially sacrificing accuracy.

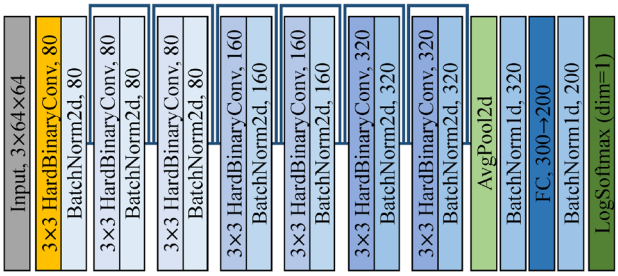

Fig. 2. Overall architecture of the proposed binarized ResNet14-Wide for Tiny ImageNet (64×64 input). The network stacks 3×3 HardBinaryConv layers followed by BatchNorm2d across three stages with widths 80/160/320, and uses global average pooling and a Batch Normalization-Fully Connected-Batch Normalization (BN–FC–BN) classification head.

To mitigate the severe information loss introduced by binarization, we insert Batch Normalization (BN) after each binarized convolution. BN stabilizes the activation distribution by normalizing per-channel statistics and provides learnable scale and bias parameters (γ, β), which compensate for the limited dynamic range of binary features. This normalization-and-affine transformation improves gradient flow and training stability, enabling the binarized ResNet14-Wide to better preserve discriminative information while keeping the overall computation lightweight.

In our prior studies, we found that a binarized ResNet fails to converge when Rectified Linear Unit (ReLU) [21] is used in the BasicBlock as in the original architecture. ReLU outputs in $[0, +\infty)$, producing a positive mean and depth-amplified variance that misaligns with the symmetric ±1 activation/weight assumption, thereby accumulating quantization error and scale drift, and destabilizing training. Given that binary networks are already information-limited, ReLU removes all negative values and effectively truncates gradient pathways, leading to systematic gradient vanishing.

We previously used HardTanh [22] to clamp outputs to $[-1,1]$. Although this "revived" the network and enabled learning, the accuracy was suboptimal. We therefore propose replacing both ReLU and HardTanh with HardSwish [23]. HardSwish preserves small negative outputs on $(-3,0)$ and provides a broad, continuous region with non-zero gradients around the origin, which better matches binary modeling and stabilizes training. Its definition is:

$$\text{HardSwish}(x) = x \times \frac{\text{ReLU6}(x+3)}{6} f(x) = \begin{cases} 0, & x \le -3 \\ \frac{x(x+3)}{6}, & -3 < x < 3 \\ x, & x \ge 3 \end{cases} \quad (1)$$

The derivative is:

$$\text{HardSwish}'(x) = \begin{cases} 0\,, & x \le 3 \\ \frac{2x+3}{6}\,, & -3 < x < 3 \\ 1\,, & x \ge 3 \end{cases} \quad (2)$$

Through experiments, we found that compared with the ResNet14 model using the HardTanh function, applying HardSwish in the BasicBlock can significantly improve model performance, as shown in Table III. The error rates in this table are the averages of the best accuracies using the Tiny ImageNet datasetover ten training runs, and a t-test was conducted to confirm the statistical significance of the improvement.

TABLE III. PERFORMANCE COMPARISON BETWEEN HARDTANH AND HARDSWISH IN RESNET14-WIDE

Model (ResNet14-Wide)	Tiny ImageNet Top1 Error	CIFAR-100 Top1 Error
Using HardTanh	56.60 ± 0.22%	42.86 ± 0.38%
Using HardTanh	47.58 ± 0.28%	29.81 ± 0.19%

Compared with ReLU, gradients exist almost everywhere on (−3,3). This overlaps more with the STE window commonly used in binary networks: the negative region is no longer "cut off", early training avoids large dead zones, and gradient density is healthier. In addition, HardSwish is a piecewise-linear approximation that is quantization/integer-implementation friendly—one reason it was adopted in MobileNetV3.

C. The "Sign Flip" Issue in Binarized Networks

In binarized neural networks, weights and activations are constrained to $\{-1, +1\}$, which aggressively compresses magnitude information and elevates the sign itself to the primary discriminative carrier; as a result, any attention gate injected into the backbone ceases to be a gentle intensity dimmer and instead behaves more like a hard operation that shifts the decision threshold or flips signs, so the choice of output range $[-1,1]$ versus $[0,1]$ is

not a trivial numeric preference but a matter of operator semantics and training dynamics: the former can introduce per-element sign inversion while the latter can induce hard masking and gradient blockage; compounding this, both gate types are tightly entangled with batch normalization, STE, residual paths, and the placement of attention relative to the binarization Sign(), a small misstep can trigger train deploy mismatch, drifting statistics, oscillatory learning curves, or even accuracy collapse.

Placed at the end of the block and before the next Sign(), a [0,1] gate scales X without immediate inversion but drives many entries toward zero. Although $\mathrm{Sign}(X')$ still matches $\mathrm{Sign}(X)$ at this moment, those near-zero values become extremely sensitive to tiny BN shifts, so small drift can trigger widespread flips and unstable training and validation. If the gate is hardened into a binary mask, it also risks ternary intermediates and train–deploy inconsistency.

At the same placement, a [−1,1] multiplicative gate yields $X'' = \mathrm{Attention}_{[-1,1]} \odot X$, and $Sign(X'') \neq \mathrm{Sign}(X)$ exactly where $A[-1,1] < 0$. Such unconstrained, data-dependent negative gating can behave like noisy per-element threshold tuning, which often correlates with BN-statistic drift and training instability.

Importantly, threshold tuning itself is not inherently harmful; what matters is whether it is controlled and selective. Rather than relying on potentially noisy negative multiplicative gates, we aim for a mechanism that enables limited, high-quality threshold tuning by adjusting the binarization boundary in a stable and interpretable manner (e.g., via a bounded threshold τ with controllable strength δ).

III. Threshold-Tuning Coordinate Attention for Binary Neural Networks

A. Coordinate Attention (CA)

Coordinate Attention first performs 1-D global average pooling for each channel, separately along the width (W) and the height (H). Intuitively, x_h preserves vertical positional information (one value per row), while x_w preserves horizontal positional information (one value per column) as shown in Eq. (3). This is the origin of "Coordinate": it performs global aggregation without discarding positional cues along the other axis. Given an input feature map $x \in R^{B*C*H*W}$, the two pooling operations are formulated as

$$x_h = \mathrm{AvgPool}_H(x) \in R^{B*C*1*W}$$

$$x_w = \mathrm{AvgPool}_W(x) \in R^{B*C*H*1} \quad (3)$$

where B, C, H, and W denote the batch size, number of channels, height, and width, respectively. In this notation, the symbol "$*$" is used only to separate tensor dimensions in the order of batch, channel, height, and width, rather than to indicate scalar multiplication. Next, the two tensors are concatenated along the spatial axis and passed through a shared 1×1 convolution for channel reduction (ratio), followed by BN and ReLU, as shown in Eq. (4). At this stage, a light-weight, monotonic nonlinearity like ReLU is desirable to fuse channel information and strengthen the representation while maintaining numerical stability; ReLU is simple and efficient, and trains well in combination with BN.

$$y = \mathrm{ReLU}\big(BN(Conv_{1\times1}[x_h||x_w])\big) \quad (4)$$

Then, the feature tensor y is split back into two parts along the spatial length: g_h and g_w. Each part is passed through its own 1×1 convolution to restore the channels to C, as shown in Eq. (5).

$$g_h = \mathrm{Conv}_{1\times1}(y_h) \in R^{B*C*1*W}$$

$$g_w = \mathrm{Conv}_{1\times1}(y_w) \in R^{B*C*H*1} \quad (5)$$

A Sigmoid gate is applied to both branches to produce two axis-wise attention maps: a_h and a_w as shown in Eq. (6), constraining the weights to [0,1] for stable soft gating and to avoid amplification >1 that could cause numerical blow-up, this is also well aligned with the residual backbone.

$$a_h = \mathrm{Sigmoid}(g_h)$$

$$a_w = \mathrm{Sigmoid}(g_w) \quad (6)$$

Finally, a_h is broadcast along the width dimension and a_w along the height dimension; their pointwise product yields the final spatial attention $x_{Attention}$ as shown in Eq. (7). The resulting attention is both channel-specific (one set of weights per channel) and coordinate-aware (different for each height and width), yet computationally light since most operations occur on narrow features after 1D pooling.

$$x_{Attention} = A(h,w) = a_h(h) \times a_w(w) \quad (7)$$

The resulting attention map captures both channel and spatial information. Multiplying it back into the residual yields the attention-enhanced feature x' in Eq. (8).

$$x' = x \times x_{Attention} \quad (8)$$

The implementation of Coordinate Attention is shown in Fig. 3.

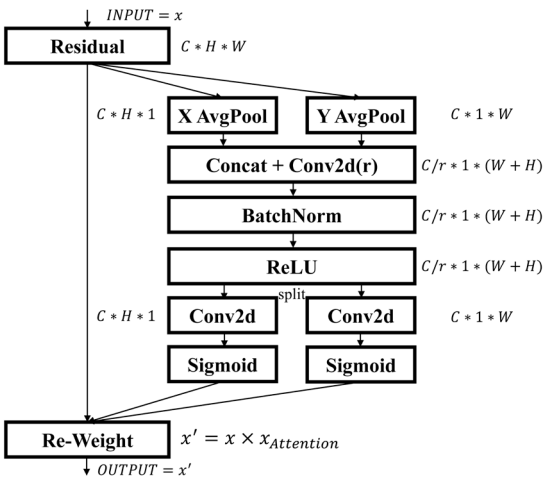

Fig. 3. Overview of the Coordinate Attention (CA) block.

B. Threshold-Tuning Coordinate Attention (TT-CA)

Compared with vanilla CA, we replace the intermediate ReLU in Threshold-Tuning Coordinate Attention (TT-CA) with HardSwish to widen the effective gradient region and make activations more near-zero–centered as shown in Eq. (9), which is friendlier to binary training and quantization.

$$y = \mathrm{HardSwish}(\mathrm{Conv}_{1\times1}[x_h || x_w]) \tag{9}$$

At the output stage, instead of Sigmoid, TT-CA adopts a Threshold-Tuning hard linear gate—a K-bit quantized variant derived from HardSigmoid. In prior work, Feature Maps Binarization (FMB) [24, 25] first lets the activation function shape the signal, then applies simple post-processing—linear scaling, clipping, or discretization—at the output to adjust the feature magnitude and zero-point. This operation effectively corresponds to a 1-bit quantization of the feature maps, mapping activations to a binary representation. In this sense, the modulation strategy of TT-CA can be regarded as an extension of the FMB-based design philosophy, since both aim to make the attention output better aligned with the characteristics of binarized representations. While FMB mainly focuses on encouraging binarized or discrete attention behavior at the output stage, TT-CA further extends this idea by introducing threshold tuning to directly modulate the sign decision of features before binarization. K-bit quantization follows this idea but extends the post-processing from 1-bit (±1) to K uniformly spaced discrete levels, preserving hardware friendliness while providing finer representational granularity and a broader, more stable gradient region for training. In addition, based on our previous findings that directly applying 1-bit convolutions in SE/CBAM-style attention brought only limited gains, TT-CA adopts lightweight full-precision convolutions in the attention branch to maintain effective threshold generation, while still constraining the output through quantization-friendly functions.

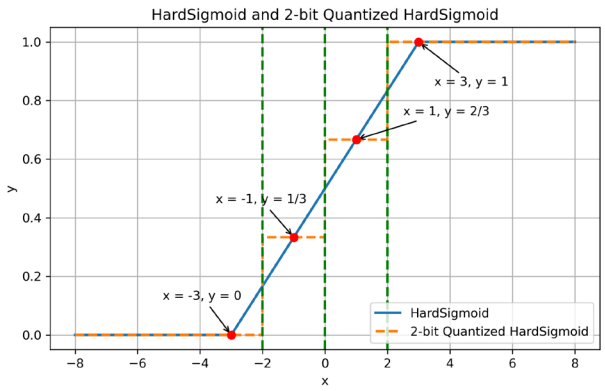

Fig. 4. Input–output curves of HardSigmoid and its 2-bit quantized version. The blue solid line represents the HardSigmoid function, while the orange dashed line denotes the 2-bit Quantized HardSigmoid. The green dashed lines indicate the threshold points used for quantization, and the red solid dots mark the four discrete output levels produced after 2-bit quantization.

In K-bit quantization, when using 2-bit as shown in Fig. 4, we adopt uniform quantization and round to the nearest grid point (rounding applied in the forward pass). In this case, three threshold points define the rounding regions, as indicated by the green dashed lines in the figure.

Compared with 1-bit binarized gating, which often degenerates into an on/off switch, the resulting attention output changes too coarsely and is prone to over-flipping or under-flipping, thus limiting its ability to recover lost information. We therefore adopt 2-bit gating as a middle ground between full-precision gating and 1-bit gating. It provides improved numerical continuity while retaining only a small set of discrete levels, enabling effective modulation without incurring the computational cost of full-precision (FP) gates.

The 2-bit quantized set for q is $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$ as shown in Eq. (10).

$$q = Quantize_k\big(HardSigmoid(a)\big) \in \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\} \tag{10}$$

To keep the attention branch lightweight and deployment-friendly, we remove the intermediate Batch Normalization (BN) in TT-CA. In low-bit backbones, the relative overhead of BN is non-negligible. Moreover, since TT-CA employs a K-bit quantized gate, additional normalization can over-smooth the gate's dynamic range and drive it toward saturation, thereby reducing effective modulation. Therefore, we replace BN with an identity mapping at the bottleneck to improve robustness and efficiency.

Moreover, since the backbone weights are binarized to −1 and 1, we further apply Eq. (11) to map the output g to $\{-1, -\frac{1}{3}, \frac{1}{3}, 1\}$.

$$g = 2q - 1 \in \left\{-1, -\frac{1}{3}, \frac{1}{3}, 1\right\} \tag{11}$$

g_h and g_w are obtained from the strip (axis-wise) pooling and the input-conditioned branches, respectively. Their gating is controlled by the threshold parameter δ. τ is computed by Eq. (12). We refer to τ as the threshold.

$$\tau = \frac{1}{2}\delta(g_h + g_w) \tag{12}$$

In the end, we obtain τ as a Threshold-Tuning form of attention. In conventional CA, a multiplicative gate (amplitude scaling) is typically used, and the attention map is multiplied back into the input. In contrast, TT-CA uses an additive threshold gate, as in Eq. (13).

$$x' = x - \tau \tag{13}$$

Binarization is essentially a comparator: $\mathrm{sign}(x)$ is equivalent to comparing x against 0. After introducing τ, it becomes $\mathrm{sign}(x - \tau)$, a comparison against τ. Thus, τ serves as an adaptive, coordinate-aware, input-dependent reference line. Strong activations (large $|x|$) are largely unaffected; only borderline responses (near 0) are selectively flipped—this is precisely the "selectivity" of attention. Fig. 5 shows the architecture of TT-CA.

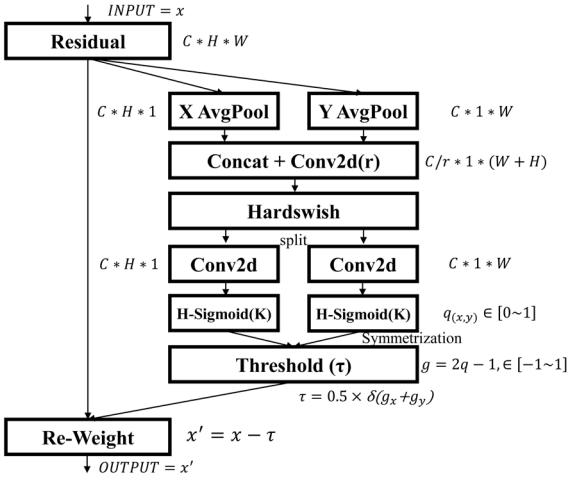

Fig. 5. Overview of the Threshold-Tuning Coordinate Attention (TT-CA) module.

IV. Experiment

A. Experimental Settings

In this study, we evaluate our model on the CIFAR-100 and Tiny ImageNet datasets.

CIFAR-100 follows the same 32×32 color images, but it contains 100 classes with 600 images per class, of which 500 are used for training and 100 for validation.

The Tiny ImageNet dataset is composed of 64×64 color images and contains 100 classes, each with 500 training images and 50 validation images.

We employ these two datasets to assess the deployability of our model under different input resolutions.

The model backbone is based on a HardSwish-enhanced ResNet14-Wide architecture (with channel dimensions of 80, 160, and 320 across stages), upon which various attention mechanisms are integrated to verify the effectiveness of the proposed Threshold-Tuning Coordinate Attention.

We further apply standard data preprocessing and normalization to the input images. For CIFAR-100, we adopt RandomCrop(32, padding=4) to enhance generalization, followed by RandomHorizontalFlip() to improve robustness to directional variations. The images are then normalized using the dataset-specific channel mean [0.5071,0.4867,0.4408] and standard deviation [0.2675,0.2565,0.2761].

For Tiny ImageNet, we employ RandomResizedCrop(64) to randomly crop and resize each image to 64×64, and apply RandomHorizontalFlip() for horizontal augmentation. Normalization is performed using the dataset statistics with mean [0.4802,0.4481,0.3975] and standard deviation [0.2770,0.2691,0.282].

For both datasets, no augmentation is applied to the test set; only normalization is used to ensure stable and consistent evaluation.

During model training, we use a batch size of 512 for CIFAR-100 and 256 for Tiny ImageNet. The optimizer is Adam, and the learning rate follows a step-wise decay schedule: 1×10^{-3} for the first 50 epochs, 1×10^{-4} for epochs 50–70, and 5×10^{-5} for epochs 70–80. The total number of training epochs is 80, and each setting is repeated over 10 runs.

The experiments are conducted on an NVIDIA RTX 4090 GPU with 24 GB of memory.

The software environment is based on Windows 11 using an Anaconda setup, equipped with Python 3.7.16 and the GPU-enabled version of PyTorch 1.10.1.

B. Applying TT-CA to ResNet14-Wide

From Eqs. (10)–(12), it can be observed that the value of g is determined by the quantized output q, taking values from the set $\{-1, -1/3, 1/3, 1\}$. Consequently, the term $g_h + g_w$ in Eq. (12) lies within the range $[-2, 2]$. We divide this sum by 2 to scale the output into $[-1,1]$, which aligns with the weight domain of binary neural networks whose weights are constrained to $\{-1,1\}$.

The parameter δ directly controls the final threshold τ, restricting it to the interval $[-\delta, \delta]$. Since δ should not exceed 1, we explore values between 0.05 and 0.5 in our experiments, testing one value every 0.05.

We integrate the proposed TT-CA module into every BasicBlock of the network.

Specifically, TT-CA is inserted after the second convolution and before the residual addition, which corresponds to the "bottleneck" location where the feature representation is most sensitive to binarization.

At this position, TT-CA performs threshold correction to stabilize the binary activation behavior. Fig. 6 illustrates how the TT-CA module is incorporated into each BasicBlock.

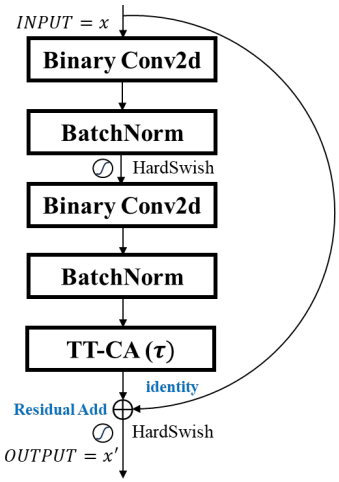

Fig. 6. Application of the proposed Threshold-Tuning Coordinate Attention (TT-CA) module inside the binary ResNet BasicBlock, inserted after the second binary convolution and before the residual addition.

We evaluate the proposed TT-CA module on the CIFAR-100 and Tiny ImageNet datasets. The TT-CA module is integrated into every BasicBlock of the ResNet architecture to examine its influence on model performance under different δ settings. We find that δ = 0.3 or 0.4 yields the best results. Tables IV and V show the error-rate reductions together with the corresponding 95%

confidence intervals, while Fig. 7 illustrates the overall accuracy curves under different δ settings.

To further assess the statistical significance of the observed error-rate reductions, Welch's t-tests were performed between the proposed TT-CA and the baseline ResNet model. The obtained p-values were 8.76×10^{-14} on Tiny ImageNet and 2.98×10^{-8} on CIFAR-100, both far below the 0.05 significance level, indicating that the error-rate reductions are statistically significant.

TABLE IV. PERFORMANCE OF RESNET14-WIDE WITH THE THRESHOLD-TUNING COORDINATE ATTENTION MODULE UNDER DIFFERENT δ VALUES ON THE TINY IMAGENET DATASET

δ values	Tiny ImageNet Top1 Error	95% CI
Baseline	47.58% ± 0.28%	[47.3765,47.7755]
0.05	46.84% ± 0.3%	[46.6261, 47.0579]
0.1	46.08% ± 0.3%	[45.8630, 46.2890]
0.15	45.82% ± 0.24%	[45.6489, 45.9991]
0.2	45.58% ± 0.28%	[45.3835, 45.7824]
0.25	45.75% ± 0.27%	[45.5557, 45.9423]
0.3	45.13% ± 0.26% (best)	[44.9425, 45.3179]
0.35	45.19% ± 0.25%	[45.0126, 45.3734]
0.4	45.22% ± 0.25%	[45.0366, 45.3961]
0.45	45.50% ± 0.38%	[45.2296, 45.7744]
0.5	45.49% ± 0.28%	[45.2913, 45.6963]

TABLE V. PERFORMANCE OF RESNET14-WIDE WITH THE THRESHOLD-TUNING COORDINATE ATTENTION MODULE UNDER DIFFERENT δ VALUES ON THE CIFAR-100 DATASET

δ values	CIFAR-100 Top1 Error	95% CI
Baseline	29.81% ± 0.19%	[29.6762, 29.9498]
0.05	29.44% ± 0.19%	[29.3074, 29.5726]
0.1	29.22% ± 0.41%	[28.9280, 29.5160]
0.15	29.21% ± 0.30%	[28.9280, 29.5160]
0.2	28.90% ± 0.21%	[28.7518, 29.0482]
0.25	28.89% ± 0.27%	[28.7008, 29.0852]
0.3	28.94% ± 0.36%	[28.6841, 29.1959]
0.35	28.94% ± 0.24%	[28.7706, 29.1174]
0.4	28.87% ± 0.25% (best)	[28.6880, 29.0420]
0.45	28.95% ± 0.30%	[28.7347, 29.1693]
0.5	29.07% ± 0.21%	[28.9157, 29.2143]

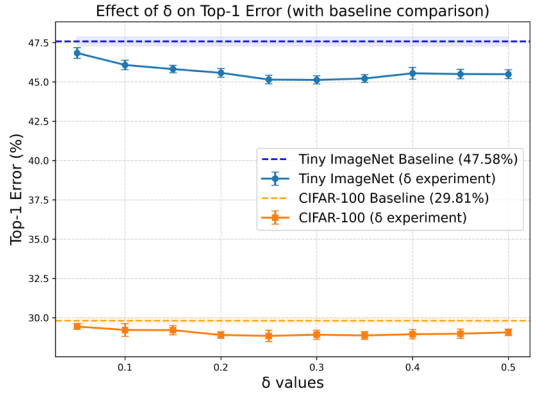

Fig. 7. Evaluation of the TT-CA module on CIFAR-100 and Tiny ImageNet. The TT-CA module is integrated into each BasicBlock of ResNet14-Wide to investigate its effect on model performance under different δ values.

Compared with the baseline model without attention, TT-CA achieves an average in Top-1 error reduction of 2.45% on Tiny ImageNet.

Although the average best improvement on CIFAR-100 is relatively modest (0.94%), it still represents a clear enhancement compared with SE, CBAM, and CA, whose original versions lead to performance degradation when applied to binary networks.

C. *Layer-Wise Evaluation of the TT-CA Module*

In addition to applying TT-CA to every BasicBlock, we also conduct experiments in which the TT-CA module is inserted into individual layers of different depths within the network.

The objective of this section is to investigate the feasibility and effectiveness of TT-CA when applied to low-level, mid-level, and high-level feature representations.

Specifically, TT-CA is independently placed in the following locations:

Layer1 (shallow stage): where feature maps contain low-level edge and texture information with relatively small receptive fields.

Layer2 (middle stage): where features become more abstract, capturing part-level and semantic cues.

Layer3 (deep stage): where high-level semantic structure is dominant, and binarization tends to introduce more severe information loss.

By isolating TT-CA to a single stage at a time, we assess how its Threshold-Tuning mechanism behaves under different feature dimensionalities and spatial resolutions.

This allows us to determine whether TT-CA is more effective at mitigating quantization-related distortions in early layers (low-dimensional features) or in deeper layers where binary activations typically suffer from higher information degradation.

Figs. 8 and 9 show the model performance on the Tiny ImageNet and CIFAR-100 datasets when applying different δ values to TT-CA at different network layers. We can clearly observe that when TT-CA is applied to a single layer only, a larger δ (i.e., stronger attention τ) is required to obtain noticeable performance gains. This is because the errors introduced by binarization are accumulated across different depths of the network: both activation binarization and weight binarization introduce quantization errors, which are gradually amplified from shallow to deeper layers.

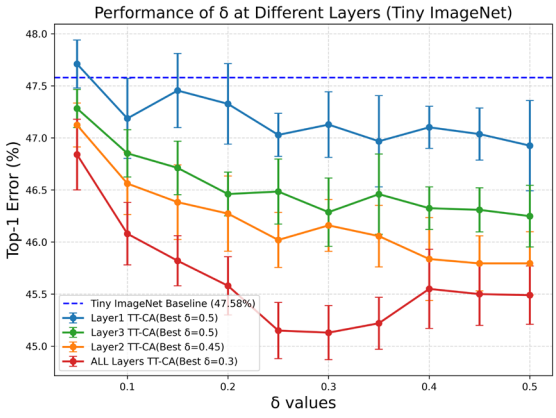

Fig. 8. Layer-wise performance on Tiny ImageNet using the TT-CA module with varying δ values applied at different depths of the network.

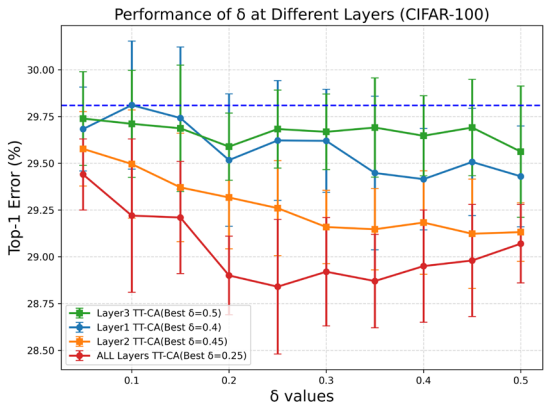

Fig. 9. Layer-wise performance on CIFAR-100 using the TT-CA module with varying δ values applied at different depths of the network.

Therefore, if TT-CA is applied only at Layer1, it can compensate for the early-stage errors, but additional quantization errors are still introduced by the subsequent layers. Conversely, if TT-CA is applied only at Layer3, it can only correct the final-stage errors, while the errors accumulated in the earlier layers are already difficult to fully compensate. As a result, when TT-CA operates at a single layer, a larger δ is needed to counteract the accumulated quantization error and to strengthen the effect of the threshold correction.

This also explains why, when TT-CA is applied to all layers simultaneously, a smaller δ is sufficient, and excessively large δ values can even degrade the performance of the attention module.

V. Discussion

Table VI compares several attention plug-ins on the same ResNet14-Wide backbone and highlights the accuracy–parameter trade-off on Tiny ImageNet in the low-bit setting. First, conventional SE and CBAM consistently degrade performance on Tiny ImageNet (48.84% and 48.40% Top-1 error) despite adding a large number of parameters (+0.539 M, ~12% over the 4.401 M baseline). This suggests that SE-like multiplicative reweighting—effective in FP models—does not transfer well to binarized/low-bit features, where the modulation granularity is limited and the relative overhead becomes more noticeable in a compact network.

TABLE VI. Accuracy–Parameter Trade-off of Attention Modules on ResNet14-Wide (Tiny ImageNet)

Attention Module (ResNet14-Wide)	#Param.	ΔParam.	Tiny ImageNet Top1 Error
None	4.401 M	0	47.58% ± 0.28%
SE	4.940 M	0.539 M	48.84% ± 0.31%
CBAM	4.940 M	0.539 M	48.40% ± 0.28%
CA	5.213 M	0.812 M	47.38% ± 0.30%
TT-CA	5.210 M	0.809 M	45.13% ± 0.26% (best)
TT-CA to Layer1	4.439 M	0.038 M	46.93% ± 0.43%
TT-CA to Layer2	4.555 M	0.154 M	45.80% ± 0.26%
TT-CA to Layer3	5.017 M	0.616 M	46.25% ± 0.30%

In contrast, CA slightly improves over SE/CBAM (47.38% error) but remains close to the baseline while introducing the largest overhead (+0.812 M, ~18%). Our TT-CA achieves the best result (45.13% ± 0.26%, at the best δ) with a similar parameter increase to CA (+0.809 M), indicating that shifting the binarization decision boundary is a more effective mechanism than standard continuous reweighting under low precision. The improvement over the "None" baseline is substantial (−2.45% in Top-1 error) while maintaining comparable model size to CA.

The layer-wise variants further show that where TT-CA is inserted matters as much as how it is designed. Applying TT-CA only to Layer1 yields a modest gain (46.93%) with negligible overhead (+0.038 M), while Layer2 provides a better accuracy–cost balance (45.80% with only +0.154 M). Interestingly, inserting TT-CA only at Layer3 increases parameters sharply (+0.616 M) but does not match the best performance (46.25%), implying that late-stage threshold tuning alone cannot fully compensate for quantization errors accumulated in earlier stages. Overall, these results support two conclusions: TT-CA is better suited than SE/CBAM/CA for binarized backbones, and Layer2 is the most cost-effective insertion point, whereas applying TT-CA across all stages maximizes accuracy at the expense of a larger parameter overhead.

We also compare the inference efficiency of different attention modules (forward pass only), as shown in Tables VII and VIII.

TABLE VII. Inference Efficiency Comparison of Attention Modules on Nvidia RTX 4090 24 GB (Forward-Only)

Attention Module on RTX4090	Peak Mem (MB)	Latency P50 (ms/img)	Latency P95 (ms/img)	Throughput (img/s)
None	88.4	3.0277	3.1453	329.214
SE	90.2	3.7226	3.8237	268.162
CBAM	90.2	5.3726	5.6228	185.900
CA	91.3	4.7319	4.8589	211.000
TT-CA	91.3	5.4170	5.5470	184.358

TABLE VIII. Inference Efficiency Comparison of Attention Modules on Nvidia Jeston Orin Nano 8 GB (Forward-Only)

Attention Module on Jeston	Peak Mem (MB)	Latency P50 (ms/img)	Latency P95 (ms/img)	Throughput (img/s)
None	86.9	14.2546	18.1139	67.198
SE	88.7	17.4228	21.6978	55.213
CBAM	88.7	21.9178	26.1232	44.196
CA	89.8	20.8666	24.5945	46.800
TT-CA	89.8	22.7498	25.8637	43.229

Peak Mem (MB) denotes the peak GPU memory usage (in MB) during the test. It reflects the maximum memory consumed by the model in the forward pass, including intermediate feature maps, temporary tensors, and workspace, and indicates whether the model can fit on memory-constrained devices and how much memory headroom remains.

Latency P50 (ms/img) is the median (50th percentile) per-image inference latency. It represents the typical/most common inference speed.

Latency P95 (ms/img) is the 95th percentile per-image inference latency. It indicates tail latency and reflects stability and jitter, i.e., 95% of inference runs are faster than this value while the slowest 5% are slower.

Throughput (img/s) is the inference throughput, measured in images per second. It is typically derived from the average latency (or measured directly) and reflects overall processing capability; higher is better.

We find that attention modules noticeably increase inference latency, and the trend is consistent across devices: $None < SE < CA < CBAM \approx TT-CA$. The "relative" overhead of attention is more pronounced on the RTX 4090: as the backbone convolutions run much faster on a high-end GPU, the fraction of time spent on attention blocks, often composed of lightweight operators, multi-branch computations, and element-wise operations, becomes larger, thereby amplifying the observed overhead. This also suggests that the cost of attention does not necessarily scale linearly with FLOPs, but is more related to factors such as kernel launch overhead, memory access patterns, and branching structure.

We further observe that the increase in Peak Mem is small. This indicates that attention mainly introduces additional compute/scheduling overhead rather than substantial GPU memory pressure: the extra parameters and intermediate activations of these attention modules are minor compared to the backbone, and the performance degradation is dominated by the latency introduced by additional operators rather than by memory capacity constraints.

It is also important to clarify the execution precision in our implementation. Although our network follows a binarized design and the convolutional weights are constrained to {−1, +1} during training, the current inference is performed using standard PyTorch floating-point kernels (FP32) rather than dedicated 1-bit bitwise operators (e.g., XNOR–popcount). Therefore, the efficiency results reported here reflect the overhead of the attention structures under FP32 execution. Implementing true 1-bit operators for the binarized convolutions (and potentially quantized attention) is expected to further reduce memory footprint and improve inference speed, which we consider an important direction for future work.

Finally, we note that tail latency P95 is more pronounced on Jetson, reflecting stability challenges on resource-constrained platforms, such as thermal/power limits, and CPU–GPU resource sharing and scheduling. Hence, for edge deployment, it is crucial to consider not only average performance but also tail latency P95 when evaluating real-time inference efficiency.

VI. Conclusion

In this work, we studied attention mechanisms for binarized CNNs and proposed Threshold-Tuning Coordinate Attention (TT-CA), which enhances low-bit models by shifting the binarization decision boundary rather than performing fine-grained multiplicative reweighting. TT-CA derives spatially aware threshold biases from coordinate-pooled features and induces controlled threshold tuning to recover discriminative capacity under severe quantization. On Tiny ImageNet with ResNet14-Wide, TT-CA (best δ = 0.3) reduces the Top-1 error from 47.58% ± 0.28% to 45.13% ± 0.26%, achieving a 2.45% improvement while keeping the parameter overhead comparable to CA. On CIFAR-100, TT-CA also yields consistent gains, reducing Top-1 error by 0.94% on average over the binarized baseline. Furthermore, the layer-wise study confirms that TT-CA can be deployed flexibly to balance accuracy and efficiency, with intermediate stages offering a particularly favorable trade-off. Overall, TT-CA provides a practical attention–binarization co-design for mobile and embedded vision.

The scope of this study is therefore limited to medium-difficulty image classification tasks using compact binarized models. Because the proposed model keeps a small parameter budget, its capacity may be insufficient for very large-scale datasets or highly diverse recognition tasks, where richer feature representations are usually required. The reported results should thus be interpreted as evidence for small-model deployment scenarios rather than as a general claim for large-scale recognition.

Future work will attempt to validate TT-CA in dedicated BNN models and specialized binary-network frameworks, while also further reducing attention overhead, implementing true 1-bit inference, and extending the evaluation to additional backbones and deployment settings.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

S. Wu: Conceived and designed the TT-CA approach, performed the experiments, conducted data analysis, and prepared the initial manuscript draft. H. Yamauchi: Provided conceptual guidance, supervised the overall research process, and reviewed and edited the manuscript. All authors have read and approved the final manuscript version.

Acknowledgment

S. Wu thanks Prof. Hiroyuki Yamauchi for his valuable guidance and constructive suggestions throughout this work. S. Wu also thanks the members of the Yamauchi Laboratory at Fukuoka Institute of Technology for helpful discussions and support.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 25, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint, arXiv:1704.04861, 2017.

[4] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," arXiv preprint, arXiv:1510.00149, 2015.

[6] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.
[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. European Conference on Computer Vision*, 2016.
[8] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
[9] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. the European Conference on Computer Vision (ECCV)*, 2018.
[10] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13713–13722.
[11] R. Sun, W. Zou, and Y. Zhan, "'Ghost' and attention in binary neural network," *IEEE Access*, vol. 10, pp. 60550–60557, 2022. doi: 10.1109/ACCESS.2022.3181192
[12] A. Krizhevsky. (2009). Learning multiple layers of features from tiny images. Tech. Rep., Univ. of Toronto. [Online]. Available: https://cave.cs.toronto.edu/kriz/learning-features-2009-TR.pdf
[13] Y. Le and X. Yang. (2015). Tiny ImageNet visual recognition challenge. CS231N Course Project Report, Stanford University. [Online]. Available: https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf
[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[15] J. Wu, Q. Zhang, and G. Xu. (2017). Tiny ImageNet challenge. CS231n Course Project Report, Stanford University. [Online]. Available: https://cs231n.stanford.edu/reports/2017/pdfs/930.pdf
[16] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *Advances in Neural Information Processing Systems*, 28, 2015.
[17] Z. Liu *et al*., "Bi-Real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm," in *Proc. the European Conference on Computer Vision (ECCV)*, 2018, pp. 722–737.
[18] H. Qin *et al.*, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, 107281, 2020.
[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," arXiv preprint, arXiv:1308.3432, 2013.
[20] S. Zagoruyko and N. Komodakis, "Wide residual networks," arXiv preprint, arXiv:1605.07146, 2016.
[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.
[22] I. Hubara *et al*., "Binarized neural networks," *Advances in Neural Information Processing Systems*, 29, 2016.
[23] A. Howard *et al.*, "Searching for MobileNetV3," in *Proc. the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.
[24] S. Wu and H. Yamauchi, "A lightweight binarized convolutional block attention module: B-CBAM," *Journal of Advances in Information Technology*, vol. 16, no. 5, pp. 751–759, 2025.
[25] S. Wu and Y. Hiroyuki, "A speed-up channel attention technique for accelerating the learning curve of a binarized Squeeze-and-Excitation (SE) based ResNet model," *Journal of Advances in Information Technology*, vol. 15, no. 5, pp. 565–571, 2024.