Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.027085

ARTICLE

Activation Redistribution Based Hybrid Asymmetric Quantization Method of Neural Networks

Lu Wei, Zhong Ma[*] and Chaojie Yang

R&D Innovation Center, Xi'an Microelectronics Technology Institute, Xi'an, 710065, China

*Corresponding Author: Zhong Ma. Email: mazhong@mail.com

Received: 21 October 2022 Accepted: 19 April 2023 Published: 22 September 2023

ABSTRACT

The demand for adopting neural networks in resource-constrained embedded devices is continuously increasing. Quantization is one of the most promising solutions to reduce computational cost and memory storage on embedded devices. In order to reduce the complexity and overhead of deploying neural networks on Integer-only hardware, most current quantization methods use a symmetric quantization mapping strategy to quantize a floating-point neural network into an integer network. However, although symmetric quantization has the advantage of easier implementation, it is sub-optimal for cases where the range could be skewed and not symmetric. This often comes at the cost of lower accuracy. This paper proposed an activation redistribution-based hybrid asymmetric quantization method for neural networks. The proposed method takes data distribution into consideration and can resolve the contradiction between the quantization accuracy and the ease of implementation, balance the trade-off between clipping range and quantization resolution, and thus improve the accuracy of the quantized neural network. The experimental results indicate that the accuracy of the proposed method is 2.02% and 5.52% higher than the traditional symmetric quantization method for classification and detection tasks, respectively. The proposed method paves the way for computationally intensive neural network models to be deployed on devices with limited computing resources. Codes will be available on https://github.com/ycjcy/Hybrid-Asymmetric-Quantization.

KEYWORDS

Quantization; neural network; hybrid asymmetric; accuracy

1 Introduction

Artificial intelligence with deep convolutional neural networks has made significant breakthroughs in many fields, which will be widely used in the aerospace field, such as situational awareness [1], intelligent obstacle avoidance [2], and remote sensing image in-orbit detection [3]. The biggest challenge for applying artificial intelligence in the aerospace field is that these artificial intelligence algorithms based on deep convolutional neural networks require a lot of memory and computational cost. In order to efficiently deploy neural networks on embedded devices, several model compression methods have been widely explored. Quantization is an essential technique for adopting deep neural networks in energy- and memory-constrained devices.

This paper is focused on Integer-only quantization for inference. Quantization is a method of quantizing the high-precision parameters of the neural network into low-precision parameters in a finite set, thereby speeding up the computation. High-precision parameters have a more extensive dynamic range, so the 32-bit floating-point data type is usually used in training. After training, in order to reduce the size of the neural network algorithm, the 32-bit floating-point neural network is quantized to an 8-bit or even lower bit integer network.

How to quantize a floating-point network to an integer network requires designing a proper mapping method. Quantization usually results in a loss of accuracy due to information lost. How to improve the accuracy of the quantized neural network considering hardware efficiency is the key problem that needs to be solved. A good quantization mapping method should resolve the two following questions to improve the deployment performance.

The first question is the trade-off between the accuracy of the quantized neural network and the difficulty of deployment and implementation. The simpler the mapping strategy is, the easier and faster the deployment on embedded devices will be, but the loss of accuracy will increase. The more complex the mapping strategy is, the lower the loss of accuracy will be. However, the deployment on embedded devices will be more difficult and result in enormous computational overhead. The commonly used quantization method is symmetric quantization for easy implementation on embedded devices. This method works well only for symmetric distributions, but most distributions of the neural networks are asymmetric.

The second question is the trade-off between range and quantization resolution, which significantly influences quantization parameters' computation. The larger the clipping range is, the lower the data clipping loss will be. However, the quantization resolution will be lower. The smaller the data clipping range is, the higher the quantization resolution will be, but the data clipping loss will be greater. Range and quantization resolution affect each other, and there is no suitable method to guide how to balance them.

We propose an activation redistribution hybrid asymmetric quantization mapping method for Integer-only inference to resolve these two questions. Our contribution can be listed as follows:

Firstly, we propose a hardware-friendly hybrid asymmetric quantization method for Integer-only inference of neural networks, of which the activation uses asymmetric activation quantization and the weights use symmetric quantization. The proposed method can avoid the additional data-dependent computation, achieve higher accuracy without any computational overhead on embedded accelerators, and resolve the contradiction between the accuracy of the quantized neural network and the ease of deployment and implementation.

Secondly, we introduce an activation redistribution method to compute the quantization parameters achieving lower quantization error. This method has no restrictions on data distribution, and can get the balance between range and quantization resolution.

2 Related Works

Most of the existing quantization approaches asymmetric quantization or symmetric quantization [4]. The asymmetric quantization function is as follows:

$$r = f(Q) = s \cdot Q + D \tag{1}$$

$$Q = f^{-1}(r) = round\left(\frac{r - D}{s}\right) \tag{2}$$

where f and f^{-1} are the quantization mapping function, f^{-1} is the inverse function of f, round is the rounding operation, r is the floating point real value, Q is the integer value after quantization, s and D are quantization parameters. s is the scaling factor, and D is the zero point, chosen such that the *0* value would exactly map to quantized values.

Symmetric quantization is a simplified version of the general asymmetric case [5]. The symmetric quantizer restricts the quantization parameter D to *0* [6].

On the one hand, different quantization mapping functions are applicable for different data distributions. The data distributions of each layer in the neural network are not same. Figs. 1 and 2 illustrate the activation distributions for each convolutional layer in the Yolo-v3 tiny model. We divide the data distributions into two categories: one is approximately symmetric, as shown in Fig. 2, and the other is asymmetric, as shown in Fig. 1. Symmetric quantization is much simpler and hardware-friendly, but is only effective for symmetric distribution. The asymmetric quantization does not require the data distribution to be symmetric around zero, but it is more expressive because there is an extra quantization parameter D and a computational overhead. The activation distributions of twelve convolutional layers (layer 3, layer 5, layer 7, layer 9, layer 11, layer 13, layer 14, layer 15, layer 16, layer 21-1, layer 21-2, layer 23) are asymmetric. Only two activation distributions of convolutional layers (layer 1 and layer 19) are approximately symmetric. Most activation distributions of the Yolo-v3 tiny model for detection are asymmetric, so the traditional symmetric quantization method suffers from a considerable loss of accuracy for the small target detection tasks.

On the other hand, the quantization parameters are very important for both asymmetric and symmetric quantization and affect the performance of the quantized neural network. The quantization parameters depend on the clipping range, and the scaling factors divides the given range of real values into a number of partitions. Usually, a series of calibrations are used as the input of a neural network to compute the typical range of activations [7,8]. A straightforward choice is to use the min/max of the data for the clipping range [7], which may unnecessarily increase the range and reduce the quantization resolution. One approach is to use the i-th largest/smallest value instead of the min/max value as the clipping range [9]. Another approach is to select the clipping range by some kinds of information loss between the original real values and the quantized values [10,11], including KL divergence [12,13], Mean Squared Error (MSE) [14–17], or entropy [18]. There are other methods to get the clipping range by learning the clipping range during training, including PACT [19], LQNets [20], LSQ [21], and LSQ+ [22]. When computing the data clipping range by KL, MSE, or other methods between the original real value and the quantized value, the absolute value of the data is first taken. Therefore, the data distribution in the range of positive and negative values cannot be effectively measured, and there is a problem of wasting the dynamic range of the data. At the same time, simply and directly taking the maximum and minimum values as the clipping thresholds cannot reflect the data distribution. So for the hybrid asymmetric quantization mapping strategy, there is no suitable method to compute the clipping range.

Therefore, the data distribution is not taken consideration in the current one-size-fits-all quantization methods, and there is no guiding principle on how to choose the most suitable method to compute the clipping range, so the current quantization methods cannot adapt to different neural network structures, and perform poorly for tasks with higher accuracy requirement.

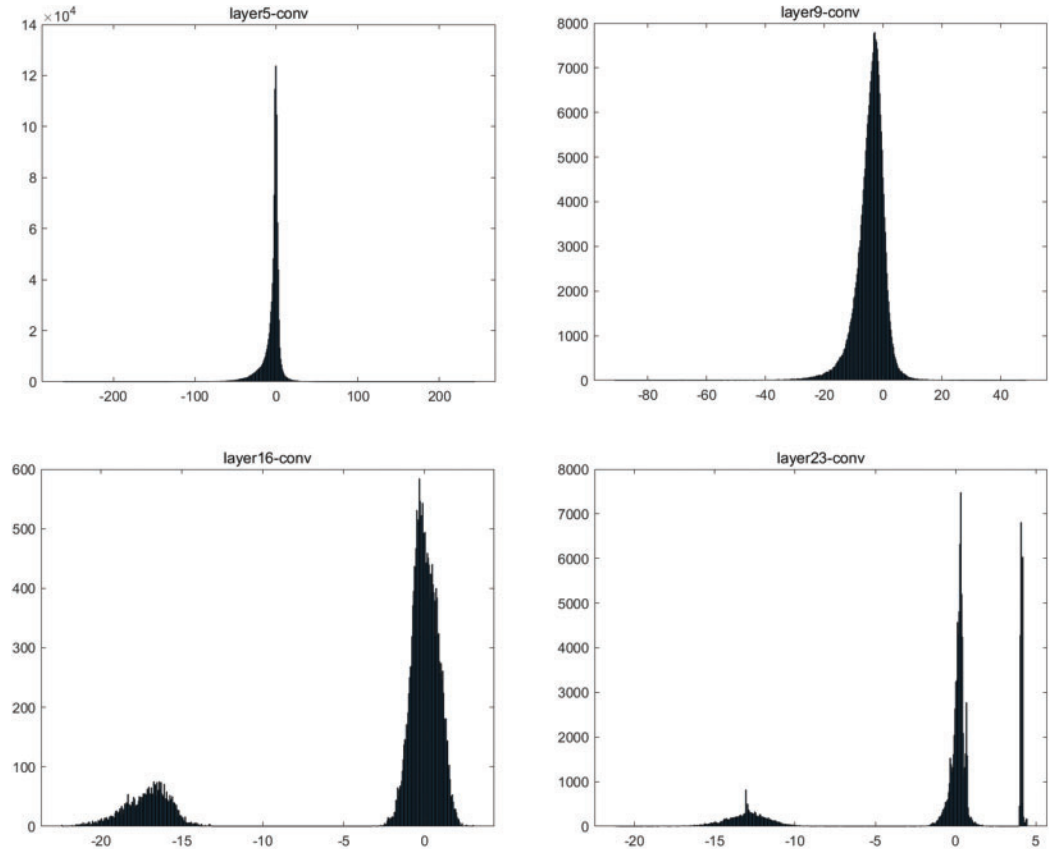

Figure 1: The activation distributions of four representative convolutional layers (layer 5, layer 9, layer 16, layer 23) of the Yolo-v3 tiny model for detection. These activation distributions are asymmetric. The horizontal axis is the activation value, and the vertical axis is the activation density

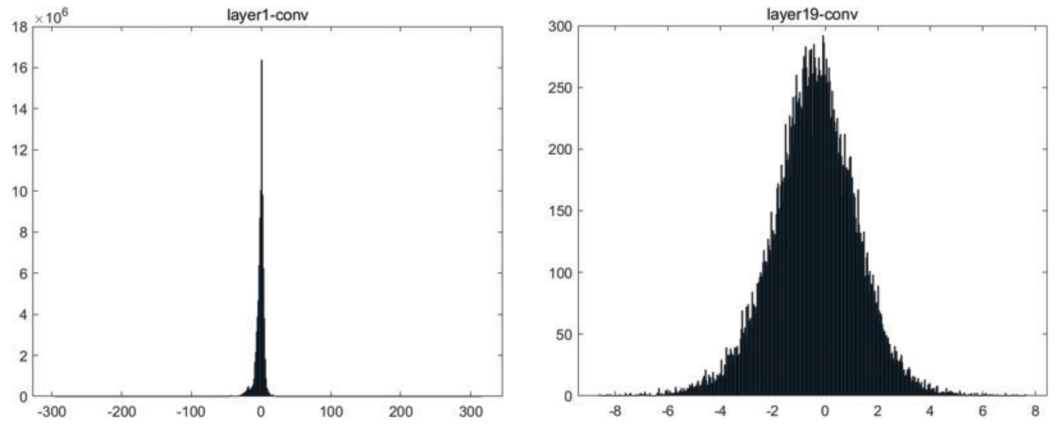

Figure 2: The activation distributions of two convolutional layers (layer 1 and layer 19) of the Yolo-v3 tiny model for detection. These activation distributions are approximately symmetric. The horizontal axis is the activation value, and the vertical axis is the activation density

3 Design

3.1 Overall Design Scheme

We propose an activation redistribution hybrid asymmetric quantization method for Integer-only inference of neural networks with simplicity and efficient implementation to hardware. The activation uses asymmetric activation quantization and the weights use symmetric quantization that avoids the additional data-dependent computation. A neural network usually consists of various layers, including the convolutional layer, the relu layer, the leaky-relu layer, the relu6 layer, the sigmoid layer, the tanh layer, and the FC layer, etc. We propose a hybrid asymmetric quantization method for neural networks and the corresponding method to compute the quantization parameters. For the computationally expensive layers, including the convolutional layer and the FC layer, we propose how to effectively quantize these layers according to the hybrid quantization parameters. For the non-linear layers, such as the relu layer, the leaky-relu layer, the relu6 layer, the sigmoid layer, etc, we propose a quantization template. All the non-linear layers can be quantized according to this template.

3.2 The Hybrid Asymmetric Integer-Only Quantization Method

In order to take into account the inference speed, accuracy, and convenience of the deployment for a quantized neural network, we propose a hybrid quantization method with asymmetric activation quantization and symmetric weight quantization. So the quantization mapping functions of the activation and weights are:

$$\text{input}^q_{i,j,k} = round\left(\frac{\text{input}^f_{i,j,k} - D_{in}}{s_{in}}\right) \tag{3}$$

$$w^q_{k,n} = round\left(\frac{w^f_{k,n}}{s_w}\right) \tag{4}$$

where $\text{input}^f_{i,j,k}$ is the activation of the neural network, $w^f_{k,n}$ represents the weights of the k-th input channel and the n-th output channel, $\text{input}^q_{i,j,k}$ is the quantized input and $w^q_{k,n}$ is the quantized weight, s_w is the quantization parameter of the weights, s_{in} and D_{in} are the quantization parameters of the convolution input.

For the computationally expensive layers, including the convolutional layer and the FC layer, we propose how to effectively quantize these layers according to the hybrid quantization parameters. The quantization of the FC layer is as same as the convolutional layer.

For the non-linear layers, such as the relu layer, the leaky-relu layer, the relu6 layer, the sigmoid layer, etc., we propose a quantization template. All the non-linear layers can be quantized according to this template. The proposed method can achieve higher accuracy without any execution time overhead on embedded accelerators.

3.2.1 The Method to Quantize the Convolutional Layer

How to quantize the convolutional layer needs to be inferred from the computational principles of the convolutional layer. The computation principle of the convolutional layer is:

$$\sum \left(\text{input}^f_{i,j,k} \cdot w^f_{k,n}\right) + \text{bias}^f_k = \text{output}^f_{l,m,n} \tag{5}$$

where bias^f_k is the k-th bias of the convolutional layer, and $\text{output}^f_{m,n,l}$ is the output of the convolutional layer. All the above data types are floating-point.

According to the computation principle of the convolution layer and the proposed hybrid asymmetric quantization strategy, how to quantize the convolutional layer can be inferred. The activations of the convolutional layer (including the input and output) adopt asymmetric quantization mapping, and the weights of the convolutional layer adopt symmetric quantization mapping. The computation principle of the quantization for the convolutional layer is:

$$\left\{\left(\frac{s_{in}\cdot s_w}{s_{out}}\cdot 2^S\right)\left(\sum\left(\frac{\mathrm{input}^f_{i,j,k}-D_{in}}{s_{in}}\cdot\frac{w^f_{k,n}}{s_w}\right)\right)+\frac{\mathrm{bias}^f_k+D_{in}\cdot\sum w^f_{k,n}-D_{out}}{s_{out}}\cdot 2^S\right\}\cdot 2^{-S}$$
$$=\frac{\mathrm{output}^f_{l,m,n}-D_{out}}{s_{out}} \tag{6}$$

where s_{out} and D_{out} are the quantization parameters of the convolution output, and S is the shift parameter for the inference process of the convolution layer.

The method to quantize the convolutional layer can be divided into 5 steps according to Eq. (6), as shown in Algorithm 1. Algorithm 1 is based on Eq. (6), and Eq. (6) illustrates how to get the integer output of the convolution layer from the integer input and the integer weights. In Eq. (6), the integer output is represented as $\frac{\mathrm{output}^f_{l,m,n}-D_{out}}{s_{out}}$, the integer input is represented as $\frac{\mathrm{input}^f_{i,j,k}-D_{in}}{s_{in}}$,and the integer weight is represented as $\frac{w^f_{k,n}}{s_w}$. $\frac{\mathrm{input}^f_{i,j,k}-D_{in}}{s_{in}}$ and $\frac{w^f_{k,n}}{s_w}$ should be on the left side of Eq. (6), $\frac{\mathrm{output}^f_{l,m,n}-D_{out}}{s_{out}}$ should be on the right side of Eq. (6), and in order to make both sides of Eq. (6) equal and involve only integer arithmetic, we can get that the left side of Eq. (6) should be $\left\{\left(\frac{s_{in}\cdot s_w}{s_{out}}\cdot 2^S\right)\left(\sum\left(\frac{\mathrm{input}^f_{i,j,k}-D_{in}}{s_{in}}\cdot\frac{w^f_{k,n}}{s_w}\right)\right)+\frac{\mathrm{bias}^f_k+D_{in}\cdot\sum w^f_{k,n}-D_{out}}{s_{out}}\cdot 2^S\right\}\cdot 2^{-S}$ on the basis of Eq. (5). The first step to quantize the convolutional layer is to compute the hybrid asymmetric quantization parameters, including s_{in}, D_{in}, s_w, s_{out} and D_{out}. The second step is to quantize the floating-point activations and weights into integer with the hybrid asymmetric quantization parameters according to Eqs. (7)–(10). The third step is to execute the multiplication and accumulation operations of the integer activations and weights, which is represented as conv in Fig. 3. The fourth step is to compute the dequantization parameters, including shift parameter S, multiplication parameter MUL, and addition parameter ADD. Dequantization is the procedure proposed in this paper to get the integer convolutional output from the results of the third step. The Dequantization procedure consists of multiplication, addition and shift. The last step is to complete the dequantization on the results of the third step in order of multiplying, adding and shifting. The multiplier is MUL. Adding uses parameter ADD, means Adding ADD. Shifting uses parameter S, means Multiplying 2^{-S}, so 2^{-S} can be implemented with an efficient bit-shift. The procedure for quantizing the convolutional layer is shown in Fig. 3.

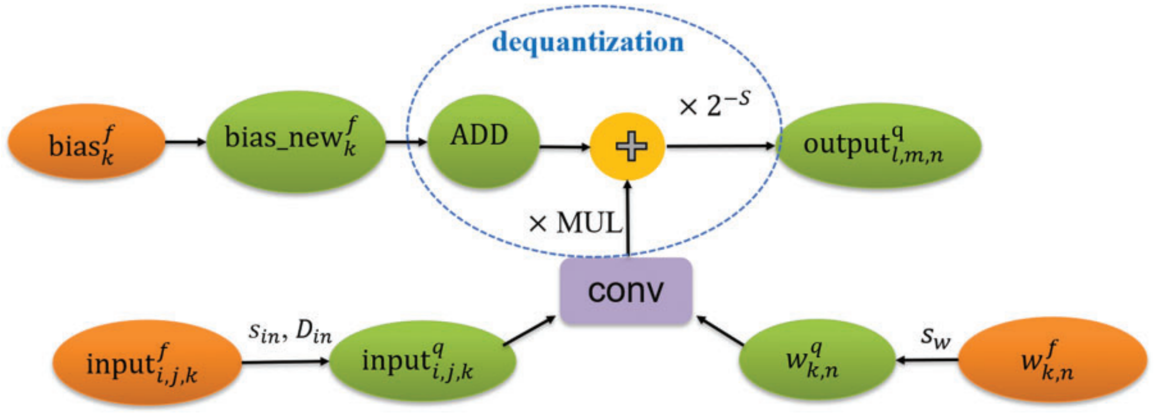

Figure 3: The procedure for quantizing the convolutional layer

Algorithm 1: Quantizing a convolutional layer

Inputs: the convolutional activation and weights
Outputs: the convolutional output
1. Compute quantization parameters. How to get the hybrid asymmetric quantization parameters is shown in Section 3.3.
2. Quantize the floating-point inputs and weights of the convolutional layer to integers.

$$\text{input}_{i,j,k}^f = MAX\left(MIN\left(\text{input}_{i,j,k}^f, max_{in}\right), min_{in}\right) \tag{7}$$

$$w_{k,n}^f = MAX\left(MIN\left(w_{k,n}^f, max_{out}\right), min_{out}\right) \tag{8}$$

$$\text{input}_{i,j,k}^q = round\left(\frac{\text{input}_{i,j,k}^f - D_{in}}{s_{in}}\right) \tag{9}$$

$$w_{k,n}^q = round\left(\frac{w_{k,n}^f}{s_w}\right) \tag{10}$$

3. Execute the multiplication and accumulation operations of the integer activations and weights.
4. Compute dequantization parameters, including shift parameter S, multiply parameter MUL, and add parameter ADD.

$$S = floor\left(-\log 2\left(\frac{s_{in} \cdot s_w}{s_{out}}\right)\right) + (8-1) \tag{11}$$

$$MUL = round\left(\frac{s_{in} \cdot s_w}{s_{out}} \cdot 2^S\right) \tag{12}$$

$$bias\_new_k^f = bias_k^f + D_{in} \cdot \sum w_{k,n}^f \tag{13}$$

$$ADD = round\left(\frac{bias\_new_k^f - D_{out}}{s_{out}} \cdot 2^S\right) \tag{14}$$

5. Execute dequantization to get the convolutional output.

In step 2, $\text{input}_{i,j,k}^f$ and $\text{output}_{m,n,l}^f$ are clipped according to their respective thresholds, as shown in Eqs. (7) and (8), and then they are quantized to integer according to Eqs. (9) and (10). MAX in Eqs. (7) and (8) is to take the maximum value and MIN in Eqs. (7) and (8) is to take the minimum value.

In step 4, the shift parameter S, multiply parameter MUL, and add parameter ADD are computed according to Eqs. (11)–(14). A convolution layer has several groups of dequantization parameters, the number of dequantization parameters is the same as the number of output channels. We should convert the floating-point biases of a convolutional layer to add parameters. In order to simplify the process of inference, we modify the floating-point biases bias_k^f to bias_new_k^f, thereby accelerating the inference speed of the convolutional layer. $\sum w_{k,n}^f$ is the sum of the weights in units of output channels, that is, how many output channels there are, how many $\sum w_{k,n}^f$ values are computed.

3.2.2 The Method to Quantize the Non-Linear Layers

This section introduces how to quantize the non-linear layers. We propose a quantization template for the non-linear layers. All the non-linear layers can be quantized according to this template. We introduce how to quantize the relu layer, the leaky-relu layer, the relu6 layer, the sigmoid layer, and the tanh layer according to the proposed quantization template.

The computation principle of the nonlinear layers can be expressed as:

$$output_{l,m,n}^f = F(input_{i,j,k}^f, x) \tag{15}$$

where F is the function of a non-linear layer, x is the fixed floating-point real value. For the relu layer and the leaky-relu layer, x is 0. For the relu6 layer, x is 6. The floating-point input is represented by $\text{input}_{i,j,k}^f$, and the floating-point output is represented by $\text{output}_{l,m,n}^f$.

The quantization method of the non-linear layers is based on the lookup table. The proposed quantization template to compute the lookup table for the non-linear layers is:

$$output_{l,m,n}^q = round\left(f^{-1}\left(F\left(f\left(input_{i,j,k}^q\right), round\left(\frac{x - D_{out}}{s_{out}}\right)\right)\right)\right) \tag{16}$$

where f is the quantization mapping function as in Eq. (1), f^{-1} is the inverse function of mapping function f as in Eq. (2), the integer input is represented by $\text{input}_{i,j,k}^q$, and the integer output is represented by $\text{output}_{l,m,n}^q$.

How to use the proposed quantization template to compute the lookup tables for the relu layer, the leaky-relu layer, the relu6 layer, the sigmoid layer and the tanh layer is shown in Table 1. Negative_slope is the parameter of the leaky-relu layer.

Table 1: Quantization method for non-linear layers

Layer type	Computation principle	Proposed quantization method for non-linear layers
relu	if $\text{input}_{i,j,k}^f < 0$: $\text{output}_{l,m,n}^f = 0$; if $\text{input}_{i,j,k}^f \geq 0$: $\text{output}_{l,m,n}^f = \text{input}_{i,j,k}^f$	if $\text{input}_{i,j,k}^q < \text{Round}(-D_{in}/s_{in})$, $\text{output}_{l,m,n}^q = \text{round}(-D_{out}/s_{out})$; if $\text{input}_{i,j,k}^q \geq \text{Round}(-D_{in}/s_{in})$, $\text{output}_{l,m,n}^q = \text{round}\left(\frac{s_{in} \cdot \text{input}_{i,j,k}^q + D_{in} - D_{out}}{s_{out}}\right)$

(Continued)

Table 1 (continued)

Layer type	Computation principle	Proposed quantization method for non-linear layers
leaky-relu	if $\text{input}^f_{i,j,k} < 0$, $\text{output}^f_{l,m,n} = \text{input}^f_{i,j,k} \cdot \text{negative\_slope}$; if $\text{input}^f_{i,j,k} \geq 0$, $\text{output}^f_{l,m,n} = \text{input}^f_{i,j,k}$	if $\text{input}^q_{i,j,k} < \text{Round}(-D_{in}/s_{in})$, $\text{output}^q_{l,m,n} = \text{round}\left(\frac{(s_{in} \cdot \text{input}^q_{i,j,k} + D_{in}) \cdot \text{negative\_slope} - D_{out}}{s_{out}}\right)$; if $\text{input}^q_{i,j,k} \geq \text{Round}(-D_{in}/s_{in})$, $\text{output}^q_{l,m,n} = \text{round}\left(\frac{s_{in} \cdot \text{input}^q_{i,j,k} + D_{in} - D_{out}}{s_{out}}\right)$
relu6	if $\text{input}^f_{i,j,k} < 0$: $\text{output}^f_{l,m,n} = 0$; if $\text{input}^f_{i,j,k} \geq 0$: $\text{output}^f_{l,m,n} MIN(\text{output}^f_{l,m,n}, 6)$	if $\text{input}^q_{i,j,k} < \text{Round}(-D_{in}/s_{in})$, $\text{output}^q_{l,m,n} = \text{round}(-D_{out}/s_{out})$; if $\text{input}^q_{i,j,k} \geq \text{Round}(-D_{in}/s_{in})$, $\text{output}^q_{l,m,n} = MIN(\text{round}\left(\frac{s_{in} \cdot \text{input}^q_{i,j,k} + D_{in} - D_{out}}{s_{out}}\right), \text{round}((6 - D_{out})/s_{out}))$
sigmoid	$\text{output}^f_{l,m,n} = \frac{1}{1 + e^{-\text{input}^f_{i,j,k}}}$	$\text{output}^q_{l,m,n} = round\left(\frac{\frac{1}{1 + e^{-\left(s_{in} \cdot \text{input}^q_{i,j,k} + D_{in}\right)}} - D_{out}}{s_{out}}\right)$
tanh	$\text{output}^f_{l,m,n} = \frac{2}{1 + e^{-2 \cdot \text{input}^f_{i,j,k}}} - 1$	$\text{output}^q_{l,m,n} = round\left(\frac{\frac{2}{1 + e^{-2 \cdot (s_{in} \cdot \text{input}^q_{i,j,k} + D_{in})}} - 1 - D_{out}}{s_{out}}\right)$

3.3 The Method to Compute Quantization Parameters

This section introduces how to compute the hybrid asymmetric quantization parameters. Select several pictures as the calibration set to compute the quantization parameters for the neural network. The method to compute quantization parameters is divided into two steps. The first step is to get the clipping thresholds, and the second step is to compute the quantization parameters according to the clipping thresholds. The clipping thresholds significantly influence quantization parameters' computation.

The method to compute the clipping thresholds should balance the trade-off between range and quantization resolution. Whether the data clipping thresholds are determined by KL, MSE or other methods between the original real values and the quantized values, there is a problem of wasting the dynamic range of the data. Because these methods are on the premise that data distribution is symmetric. But most of the activation distributions are asymmetric. These methods take the absolute value of the data first when computing the clipping range, and then select the data thresholds by a certain measurement method. The operation of taking the absolute value makes these methods unable to truly reflect the data distribution both in the positive and negative range. The quantization of non-negative activations may be less effective at this point because the clipping range includes values that never appear in the input.

In order to adopt asymmetric activation distributions, and balance the trade-off between range and quantization resolution, we propose an activation redistribution method to compute the clipping thresholds achieving lower quantization error, because this method takes data distribution into consideration. The optimal clipping range for the input is $[min_{in}, max_{in}]$, the optimal clipping range for the output is $[min_{out}, max_{out}]$, the threshold of the weights is $\mathrm{th_w}$. The procedure for computing these clipping thresholds is shown in Algorithm 2 and Fig. 4.

Algorithm 2: Computing the optimal clipping thresholds

Inputs: the data distribution of activation and weights

Outputs: clipping thresholds $[min_{in}, max_{in}]$, $[min_{out}, max_{out}]$ and $\mathrm{th_w}$.

1. Transform the input activation distribution into a gaussian-like distribution by Box-Cox [23] as shown in Fig. 4. λ determines the specific type of transformation, such as square root transformation, reciprocal transformation, etc. Different distributions should choose different λ.

$$y = BC\,(\mathrm{x}) \begin{cases} \dfrac{(x+c)^{\lambda}-1}{\lambda}, & if\ \lambda \neq 0 \\ \log\,(x+c)\,, & if\,\lambda = 0 \end{cases} \tag{17}$$

2. Compute the clipping thresholds by KL divergence to get $[-max_{KL}, max_{KL}]$ [10].
3. Get $[min_{in}, max_{in}]$ by the inverse transformation as shown in Fig. 4.
4. The method to get the final output clipping range $[min_{out}, max_{out}]$ is the same as $[min_{in}, max_{in}]$.
5. The weight clipping threshold $\mathrm{th_w}$ is the maximum value of the absolute of the weights of a channel for the convolution layer. A convolution layer has several $\mathrm{th_w}$, the number of $\mathrm{th_w}$ is the same as the number of output channels.

Fig. 4 shows how to get the clipping thresholds $[min_{in}, max_{in}]$ of the input. c is a constant to ensure that the input is positive. All of the inputs plus c, and then transform the input into a gaussian-like distribution by Box-Cox, which is represented as BC in Fig. 4. d is a constant to ensure that the transformed gaussian-like distribution is symmetric. We use KL divergence to balance the trade-off between range and quantization resolution. Get the clipping thresholds $[-max_{KL}, max_{KL}]$ by KL divergence [10] of the symmetric gaussian-like distribution. In this way, the data both in the positive and negative range is taken into consideration. At last, get $[min_{in}, max_{in}]$ by the inverse transformation of Box-Cox, which is represented as BC^{-1}.

As can be seen from the above figure, when the data distribution is not symmetrical around 0, for example, the negative values are small, then the data thresholds determined by the KL divergence are not suitable, because the threshold selected for the negative value area is affected by the positive value, which cannot match the actual data distribution of negative values. The proposed method transforms an asymmetric and skewed activation distribution into a gaussian-like distribution, then get the clipping thresholds by KL divergence, and finally gets the final clipping range by the inverse transformation.

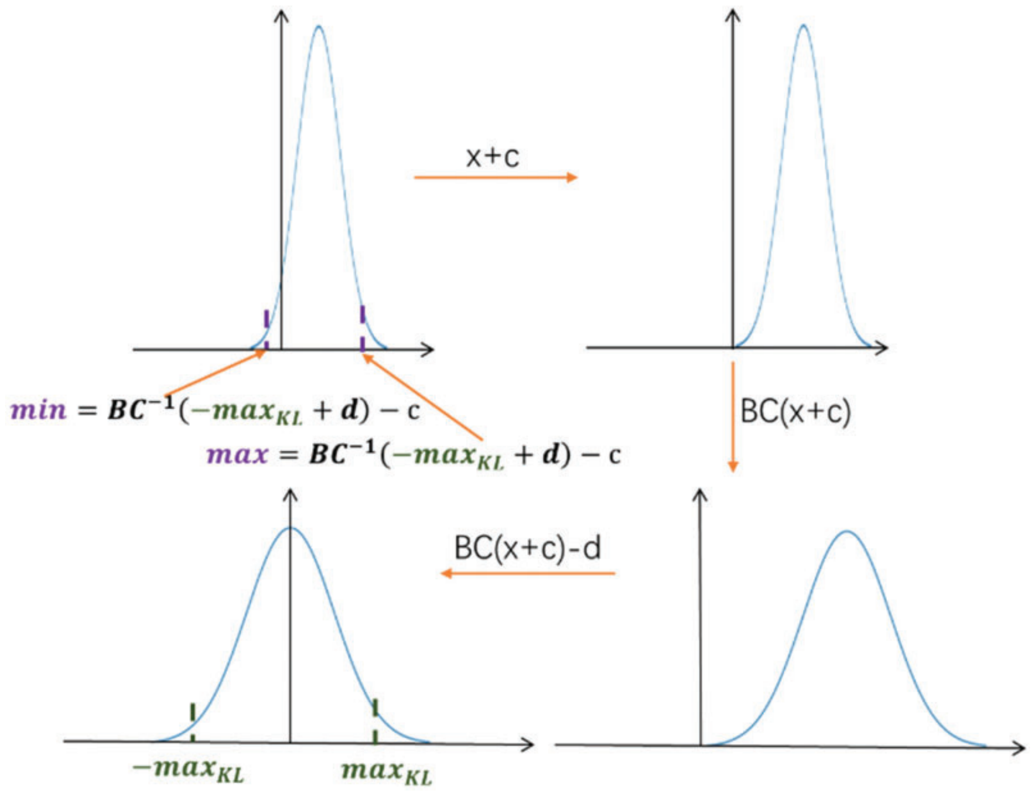

Figure 4: The activation redistribution method to compute the optimal clipping thresholds

How to compute the quantization parameters according to the clipping thresholds is as follows. The quantization parameters s_{in}, D_{in}, s_{out}, D_{out}, s_w of a layer are computed according to Eqs. (18)–(22).

$$s_{in} = \frac{max_{in} - min_{in}}{2^{bw_{in}} - 1} \tag{18}$$

$$D_{in} = \frac{min_{in} - max_{in}}{2^{bw_{in}} - 1} \cdot round\left(\frac{(2^{bw_{in}-1} - 1) \cdot min_{in} + 2^{bw_{in}-1} \cdot max_{in}}{min_{in} - max_{in}}\right) \tag{19}$$

$$s_{out} = \frac{max_{out} - min_{out}}{2^{bw_{out}} - 1} \tag{20}$$

$$D_{out} = \frac{min_{out} - max_{out}}{2^{bw_{out}} - 1} \cdot round\left(\frac{(2^{bw_{out}-1} - 1) \cdot min_{out} + 2^{bw_{out}-1} \cdot max_{out}}{min_{out} - max_{out}}\right) \tag{21}$$

$$s_w = \frac{th_w}{2^{bw_w-1} - 1} \tag{22}$$

where bw_{in}, bw_{out} and bw_w are the bit width of input, output and weight, of which 8 is commonly used.

4 Implementation and Experimental Results

The purpose of the experiments is to verify the effectiveness of the proposed hybrid asymmetric Integer-only quantization method.

4.1 Experimental Setting

The neural networks adopted in the experiments are the image classification models and the small target detection model. All of the neural networks are quantized to INT8.

Firstly, the experiments are implemented on the TIANJI NPU3.0 neural network accelerator proposed by Xi'an Microelectronics Technology Institute [24] and Cambricon MLU220 [25]. TIANJI NPU3.0 accelerator is implemented based on Xilinx ZCU102 FPGA, with self-controllable IP and application development tool chain [26]. The FPGA is Zynq UltraScale + XCZU9EG, and there are 2520 DSP slices, and the DDR4 in the programmable logic is 4 Gb. MLU220 is Based on the Cambrian MLUv02 architecture. The theoretical peak performance is 8TOPS and the power consumption is 8.25 W. These two accelerators can be widely used in edge computing scenarios to support diverse AI applications. The image classification models and the small target detection model are deployed on the neural network accelerators, and the speed and accuracy are verified. The purpose of the experiments is to verify the effectiveness of the proposed hybrid asymmetric Integer-only quantization method on embedded devices. The expected experimental results are that the proposed quantization method can improve the accuracy without affecting the speed on embedded devices compared with the traditional symmetric quantization method adopted by most embedded neural network accelerators.

Secondly, we compare the proposed method with PyTorch and NNI [27] on image classification models and the small target detection model, and the accuracy is verified on software. The CPU is Intel(R) Core(TM) i7-8700K, 3.70 GHz, and the GPU is NVIDIA GeForce GTX1070. In order to get the experimental results conveniently, software with fake-quantization [28] modules is used to simulate the accuracy on the neural network accelerator. Fake-quantization models quantization errors in the forward passes. The reason to apply fake-quantization is to quickly simulate the effects of quantization using simulated quantization operations. Codes will be available on https://github.com/ycjcy/Hybrid-Asymmetric-Quantization, from which the experimental results of the proposed method, PyTorch, and NNI can be obtained. The purpose of the experiments is to verify the effectiveness of the proposed hybrid asymmetric quantization method with the state-of-art. The expected experimental results are that the proposed hybrid asymmetric quantization method can improve the accuracy compared with PyTorch and NNI.

4.2 Dataset

The dataset for image classification application is ImageNet. ImageNet is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. The dataset has been instrumental in advancing computer vision and deep learning research.

The dataset for is small target detection application is HRSID. HRSID is a dataset for ship detection, semantic segmentation, and instance segmentation tasks in high-resolution SAR images. The dataset contains 5604 SAR images with resolutions of 0.5, 1, and 3 m.

4.3 Evaluation Metrics

In order to verify the accuracy of quantification methods extensively, we evaluate two aspects of quantization errors. One is the quantization error of a particular layer. The second is the overall quantization error of a model.

For the first aspect of quantization error, there are no ideal metrics that can perfectly measure the quantization error. Different metrics reflect the quantization error from different points. We adopt the following three metrics to measure the quantization error, including Manhattan distance, Euclidean distance, and Signal to Noise Ratio. The range of Manhattan distance and Euclidean distance is 0 to $+\infty$, and the range of Signal to Noise Ratio is $-\infty$ to $+\infty$. The smaller the Manhattan distance and the Euclidean distance are, the lower the error will be. The higher the signal-to-noise ratio is, the lower the error will be.

- Manhattan distance (sum of the absolute values of the difference between the original real values and the corresponding floating-point values after quantization):

$$d_1 = \sum_{i=1}^{i=t} |r_i - q_i| \tag{23}$$

- Euclidean distance (the square root of the sum of the square of the difference between the original real values and the corresponding floating-point values after quantization):

$$d_2 = \sqrt{\sum_{i=1}^{i=t} (r_i - q_i)^2} \tag{24}$$

- Signal to Noise Ratio:

$$SQNR = 10 \cdot \log_{10}\left(\frac{\sum_{i=1}^{i=t} r_i^2}{\sum_{i=1}^{i=t} (r_i - q_i)^2}\right) \tag{25}$$

where r_i is the original real value of floating point, q_i is the corresponding real value after quantization, and the size of the input or output data is t.

For the second aspect of quantization error, the evaluation metrics are the accuracy metrics of that model. For image classification application, we use Top-1 Accuracy (the one with the highest probability must be exactly the expected answer). For small target detection application, we use mAP (Mean Average Precision). The calculation of mAP is the same as in the internationally renowned target detection competition PASCAL VOC Challenge.

4.4 Baseline

Firstly, we use a traditional symmetric quantization method as a baseline. This method adopts symmetric quantization for both activation and weights, with the clipping range determined by KL divergence. This method is adopted by most embedded neural network accelerators, such as Nvidia's TensorRT [12], TVM [13], etc.

Secondly, as a baseline, we compare the proposed method with PyTorch and NNI on PC. PyTorch supports INT8 quantization compared to typical FP32 models allowing for a 4x reduction in the model size and a 4x reduction in memory bandwidth requirements. Hardware support for INT8 computations is typically 2 to 4 times faster compared to FP32 computing. PyTorch supports multiple approaches to quantize a deep learning model. In most cases, the model is trained in FP32 and then the model is converted to INT8. In addition, there are three functions in PyTorch to compute the clipping range. The torch.quantization.observer module in Pytorch integrates three calibration strategies, including MinMaxObserver, MovingAverageMinMaxObserver, and HistogramObserver. There is no guide on how to get the most suitable strategy. The easiest way (and the default option in Pytorch) is to directly take the minimum and maximum values by the MinMaxObserver function. The method in NNI to compute the clipping range is also to take the minimum and maximum values.

4.5 Results

4.5.1 Results on FPGA

- Classification Application

The models used for image classification are GoogleNet, MobileNetV2, and VGG16. For fair comparison and ease of reproducibility, we use well-trained models on the ImageNet dataset. For image classification application, we test Top-1 Accuracy and FPS (How many frames can be processed per second).

TIANJI NPU3.0 accelerator runs at a frequency of 200M. The resources consumption on FPGA of TIANJI NPU3.0 is shown in Table 2, including LUT, FlipFlops, Block RAMs and DSPs.

Table 2: Resources Consumption on FPGA of TIANJI NPU3.0

	Total resources	Consumption	Consumption percentage
LUT	274080	186174	67.93%
FlipFlops	548160	167977	60.64%
Block RAMs	912	547	59.98%
DSPs	2520	2048	81.27%

The results for the classification application on TIANJI NPU3.0 are shown in Table 3. A basic requirement of inference on TIANJI NPU3.0 is that it permits implementation of all arithmetic using only integer arithmetic operations, so it is a big challenge for the quantization method to reduce the accuracy loss. The proposed method ensures that all the layers of the neural network are inferenced by integer. For the three classification models, the FPS of the proposed hybrid asymmetric quantization method is the same as the FPS of the traditional symmetric quantization method. So the proposed hybrid asymmetric quantization method can improve the classification accuracy by an average of 2.02% without affecting the speed on FPGA. It can meet the accuracy requirements of image classification tasks.

Table 3: Experimental results of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for classification application on TIANJI NPU3.0

Model	PC accuracy (FP32)	Traditional symmetric quantization (INT 8)			Proposed hybrid asymmetric quantization (INT 8)		
		Top-1 accuracy	FPS	Power	Top-1 accuracy	FPS	Power
GoogleNet	67.04%	65.91%	36.63	14.44 W	**66.65%**	36.63	14.44 W
MobileNetV2	70.24%	61.54%	38.02	14.48 W	**63.79%**	38.02	14.48 W
VGG16	66.13%	62.53%	9.65	14.72 W	**65.61%**	9.65	14.72 W

The results for the classification application on MLU220 are shown in Table 4. MLU220 only supports symmetric quantization, and only convolutional layers and FC layers can be quantized to

INT8, others types of layers are all executed in FP32. The traditional method on MLU220 to compute the clipping range is to take the minimum and maximum values. We compare the proposed activation redistribution method with the traditional method on MLU220. The methods proposed in this paper are all computed offline and do not increase the computational overhead of the embedded devices, which is the same as other traditional methods. It is valid for different embedded devices,because this method proposes a new way to compute quantization parameters from a mathematical point of view, which has no effect on embedded devices.

Table 4: Experimental results of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for classification application on MLU220

Model	PC accuracy (FP32)	Traditional symmetric quantization (Only convolutional and FC layers are INT8)	Proposed activation redistribution method (Only convolutional and FC layers are INT8)
		Top-1 accuracy	Top-1 accuracy
GoogleNet	67.04%	66.97%	**67.23%**
MobileNetV2	70.24%	69.88%	**69.92%**
VGG16	66.13%	65.83%	**65.81%**

- Small Target Detection Application

The small target detection task is very challenging because the loss of accuracy is very sensitive to quantization. The model we choose for this small target detection task is Yolo-v3 tiny, a typical object detection model that has been widely adopted.

The experimental results measuring the quantization error of convolutional layers and relu layers for the small target detection application on TIANJI NPU3.0 are shown in Tables 5 and 6. It can be seen that the Manhattan distance and the Euclidean distance of the proposed hybrid asymmetric method are lower, and the signal-to-noise ratio SQNR of the proposed hybrid asymmetric method is higher.

Table 5: Comparison of quantized error of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for convolutional layers on TIANJI NPU3.0

Methods to measure quantization error	Symmetric quantization error	Proposed hybrid asymmetric quantization error
Manhattan distance	3.2468	0.3617
Euclidean distance	2.1446	0.2723
Signal to Noise Ratio	41.1668	59.0920

Table 6: Comparison of quantized error of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for RELU layers on TIANJI NPU3.0

Methods to measure quantization error	Symmetric quantization error	Proposed hybrid asymmetric quantization error
Manhattan distance	1177.0	823.5
Euclidean distance	4.0216	2.8923
Signal to Noise Ratio	38.6908	41.0279

The speed and accuracy experimental results for the small target detection application on TIANJI NPU3.0 are shown in Fig. 5 and Table 7. For the small target detection model, the FPS of the proposed hybrid asymmetric quantization method is the same as the FPS of the traditional symmetric quantization method. So the proposed hybrid asymmetric quantization method can improve the detection accuracy by 5.52% without affecting the speed on embedded devices. It can meet the accuracy requirements of small object detection tasks.

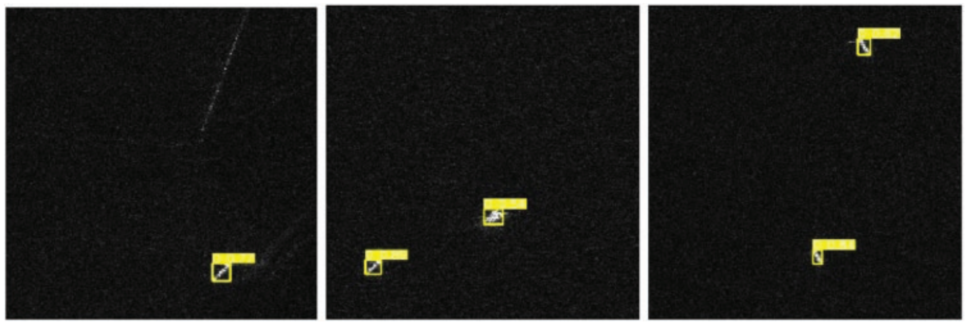
Figure 5: Small object detection on HRSID dataset

Table 7: Experimental results of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for small target detection application on TIANJI NPU3.0

Model	PC accuracy (FP32)	Traditional symmetric quantization (INT 8)			Proposed hybrid asymmetric quantization (INT 8)		
		mAP	FPS	Power	mAP	FPS	Power
Yolo-v3 tiny	90.70%	80.12%	17.89	14.48 W	**85.64%**	17.89	14.48 W

The results for the small target detection application on MLU220 are shown in Table 8. The proposed activation redistribution method can improve the detection accuracy from 82.67% to 82.83%.

Table 8: Experimental results of the proposed hybrid asymmetric quantization method and the traditional symmetric quantization method for small target detection application on MLU220

Model	PC accuracy (FP32)	Traditional symmetric quantization (Only convolutional and FC layers are INT8)	Proposed activation redistribution method (Only convolutional and FC layers are INT8)
		mAP	mAP
Yolo-v3 tiny	90.70%	82.67%	**82.83%**

4.5.2 Results on PC

- Classification Application

The models used for image classification to compare with PyTorch and NNI are the same as Section 4.5.1. The evaluation metric is Top-1 Accuracy. There are three ways in PyTorch to compute the clipping range, including MinMax, MovingAverage, and Histogram. We compare the proposed hybrid asymmetric quantization method with PyTorch and NNI, and compare the proposed activation redistribution method with MinMax, MovingAverage, and Histogram in PyTorch on fake-quantization software.

The results for the classification application are shown in Table 9 and Fig. 6. For the three classification models, the accuracy of the proposed hybrid asymmetric quantization method (66.85%, 67.38%, 66.26%) is the highest compared with PyTorch (66.50%, 66.82%, 65.22%) and NNI (66.74%, 66.50%, 65.20%). At the same time, there are three ways in PyTorch and NNI to compute the clipping range. For different models, the best strategy of PyTorch and NNI to compute the clipping range is different. The proposed activation redistribution method outperforms the three strategies of PyTorch and NNI.

Table 9: Experimental results of the proposed hybrid asymmetric quantization method, pyTorch and NNI for classification application on fake-quantization software

Model	PC accuracy (FP32)	NNI accuracy (fake INT 8)	PyTorch accuracy (fake INT 8)	Proposed hybrid asymmetric quantization accuracy (fake INT 8)			
				MinMax	Moving MinMax	Histogram	Proposed activation redistribution
GoogleNet	67.04%	66.74%	66.50%	66.82%	66.82%	66.77%	**66.85%**
MobileNetV2	70.24%	66.50%	66.82%	66.23%	66.60%	66.99%	**67.38%**
VGG16	66.13%	65.20%	65.22%	65.12%	66.20%	65.31%	**66.26%**

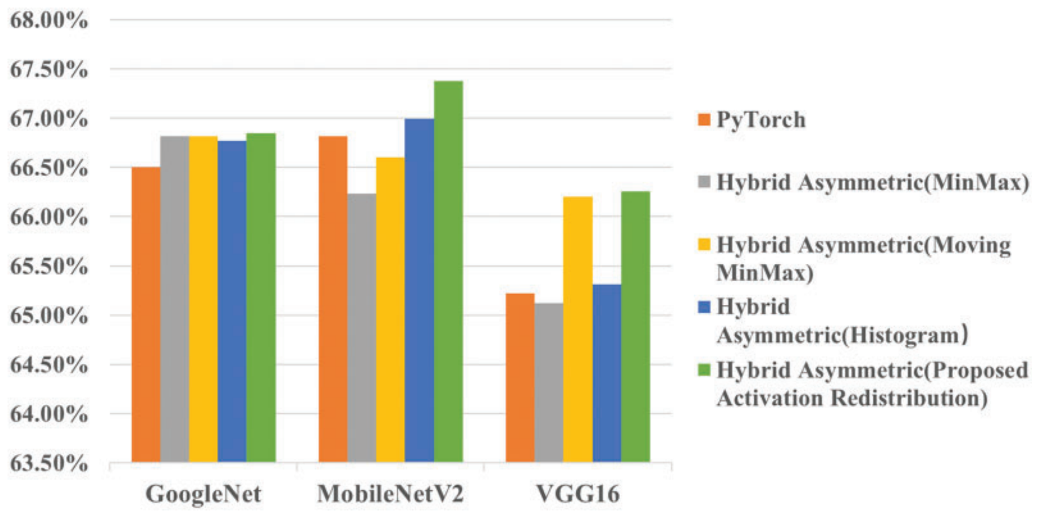

Figure 6: Comparison of the proposed method and PyTorch for image classification application. The vertical axis is the Top-1 Accuracy

- Small Target Detection Application

The model used for small target detection to compare with PyTorch is the same as Section 4.5.1. The evaluation metric is mAP. We compare the proposed hybrid asymmetric quantization method with PyTorch, and compare the proposed activation redistribution method with MinMax, MovingAverage, and Histogram in PyTorch on fake-quantization software.

The results for small target detection model application are shown in Table 10. The proposed hybrid asymmetric quantization method can improve the detection accuracy compared with PyTorch.

Table 10: Experimental results of the proposed hybrid asymmetric quantization method, pyTorch and NNI for small target detection application on fake-quantization software

Model	PC accuracy (FP32)	NNI accuracy (fake INT 8)	PyTorch accuracy (fake INT 8)	Proposed hybrid asymmetric quantization accuracy (fake INT 8)			
				MinMax	Moving MinMax	Histogram	Proposed activation redistribution
Yolo-v3 tiny	90.70%	85.7%	83.2%	86.7%	85.6%	86.1%	**90.12%**

5 Conclusion and Future Directions

We propose an activation redistribution hybrid asymmetric quantization method for Integer-only inference of neural networks. This method is suitable for both symmetric distributions and asymmetric distributions. When the proposed hybrid asymmetric Integer-only quantization method is applied to classification models, we can achieve an average accuracy improvement up to 2.02% compared with the traditional symmetric quantization method. When the proposed hybrid asymmetric Integer-only quantization method is applied to Yolo-v3 tiny model for detection, the accuracy improvement is

5.52% compared with the traditional symmetric quantization method. So, our method can make the neural networks quickly and easily deployed on the resource-constrained embedded devices.

For further work, we believe that making the distribution more friendly to quantization is a promising research direction to improve the quantization performance further.

Acknowledgement: The Authors acknowledge the support received from the Qian Xuesen Youth Innovation Foundation of China Aerospace Science and Technology Corporation under grant 2022JY51.

Funding Statement: The Qian Xuesen Youth Innovation Foundation from China Aerospace Science and Technology Corporation (Grant Number 2022JY51).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Lu Wei, Zhong Ma; data collection and experiment: Chaojie Yang; analysis and interpretation of results: Lu Wei, Chaojie Yang; draft manuscript preparation: Lu Wei, Zhong Ma. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the accessible website https://github.com/ycjcy/Hybrid-Asymmetric-Quantization.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Tang, L., Ma, Z., Li, S., Wang, Z. X. (2022). The present situation and developing trends of space-based intelligent computing technology. *Microelectronics & Computer, 39(4),* 1–8. https://doi.org/10.19304/J.ISSN1000-7180.2021.1229
2. Zhou, X. S., Wu, W. L. (2021). Unmanned system swarm intelligence and its research progresses. *Microelectronics & Computer, 38(12),* 1–7. https://doi.org/10.19304/J.ISSN1000-7180.2021.1171
3. Uçar, F., Korkmaz, D. (2020). A ship detector design based on deep convolutional neural networks for satellite images. *Sakarya University Journal of Science, 24(1),* 197–204.
4. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W. et al. (2021). A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv.2103.13630.
5. Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Baalen, M. V. et al. (2021). A white paper on neural network quantization. arXiv preprint arXiv:2106.08295.
6. Li, Y., Dong, X., Wang, W. (2020). Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. arXiv preprint arXiv:1909.13144.
7. Jacob, B., Kligys, S., Chen, B., Zhu, M. L., Tang, M. et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713. Salt Lake City, UT, USA.
8. Yao, Z. W., Dong, Z., Zheng, Z., Gholaminejad, A., Yu, J. et al. (2020). HAWQV3: Dyadic neural network quantization. arXiv preprint arXiv:2011.10680.
9. McKinstry, J. L., Esser, S. K., Appuswamy, R., Bablani, D., Arthur, J. V. et al. (2018). Discovering low-precision networks close to full-precision networks for efficient embedded inference. arXiv preprint arXiv:1809.04191.
10. Krishnamoorthi, R. (2018). Quantizing deep convolutional net-works for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.
11. Wu, H., Judd, P., Zhang, X., Isaev, M., Micikevicius, P. et al. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv preprint arXiv:2004.09602.

12. Migacz, S. (2017). 8-bit inference with TensorRT. *GPU Technology Conference*, vol. 2, pp. 7. https://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf
13. Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. *13th fUSENIXg Symposium on Operating Systems Design and Implementation (fOSDIg 18)*, pp. 578–594. Carlsbad, CA, USA.
14. Choukroun, Y., Kravchik, E., Yang, F., Kisilev, P. (2019). Low-bit quantization of neural networks for efficient inference. *ICCV Workshops*, pp. 3009–3018. Seoul, Korea.
15. Shin, S., Hwang, K., Sung, W. (2016). Fixed-point performance analysis of recurrent neural networks. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 976–980. Shanghai, China.
16. Sung, W., Shin, S., Hwang, K. (2015). Resiliency of deep neural networks under quantization. arXiv preprint arXiv:1511.06488.
17. Zhao, R., Hu, Y. W., Dotzel, J. (2019). Improving neural network quantization without retraining using outlier channel splitting. arXiv preprint arXiv:1901.09504.
18. Park, E., Ahn, J., Yoo, S. (2017). Weighted-entropy-based quantization for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5456–5464. Honolulu, Hawaii.
19. Choi, J., Zhuo, W., Venkataramani, S., Chuang, I. J., Gopalakrishnan, K. (2018). PACT: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085.
20. Zhang, D., Yang, J., Ye, D., Hua, G. (2018). LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. *European Conference on Computer Vision (ECCV)*, pp. 373–390. Munich, Germany.
21. Esser, S., McKinstry, J. L., Bablani, D., Appuswamy, R., Modha, D. S. (2019). Learned step size quantization. arXiv preprint arXiv:1902.08153.
22. Bhalgat, Y., Lee, J., Nagel, M., Blankevoort, T., Kwak, N. (2020). LSQ+: Improving low-bit quantization through learnable offsets and better initialization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 696–697.
23. Box G.E., P., Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological), 26(2),* 211–243. https://doi.org/10.1111/j.2517-6161.1964.tb00553.x
24. Jiao, F., Ma, Y., Bi, S. Y., Ma, Z. (2022). Design of instruction control system for neural network accelerator. *Microelectronics & Computer, 39(8),* 78–85.
25. Cambricon (2023). https://www.cambricon.com
26. Ma, Y., Bi, S. Y., Jiao, F., Ma, Z. (2021). A CNN accelerator with high bandwidth storage. Chinese invention patent, CN20210921363.9.
27. Microsoft (2022). Neural network intelligence (version v2.10). https://github.com/microsoft/nni
28. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M. et al. (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference. arXiv preprint arXiv:1712.05877.