remote sensing

MDPI

Article

Q-A^{2}NN: Quantized All-Adder Neural Networks for Onboard Remote Sensing Scene Classification

Ning Zhang, He Chen *, Liang Chen, Jue Wang, Guoqing Wang and Wenchao Liu

National Key Laboratory of Science and Technology on Space-Born Intelligent Information Processing, Beijing Institute of Technology, Beijing 100081, China; gqwang@bit.edu.cn (G.W.)
* Correspondence: chenhe@bit.edu.cn

Abstract: Performing remote sensing scene classification (RSSC) directly on satellites can alleviate data downlink burdens and reduce latency. Compared to convolutional neural networks (CNNs), the all-adder neural network (A^{2}NN) is a novel basic neural network that is more suitable for onboard RSSC, enabling lower computational overhead by eliminating multiplication operations in convolutional layers. However, the extensive floating-point data and operations in A^{2}NNs still lead to significant storage overhead and power consumption during hardware deployment. In this article, a shared scaling factor-based de-biasing quantization (SSDQ) method tailored for the quantization of A^{2}NNs is proposed to address this issue, including a powers-of-two (POT)-based shared scaling factor quantization scheme and a multi-dimensional de-biasing (MDD) quantization strategy. Specifically, the POT-based shared scaling factor quantization scheme converts the adder filters in A^{2}NNs to quantized adder filters with hardware-friendly integer input activations, weights, and operations. Thus, quantized A^{2}NNs (Q-A^{2}NNs) composed of quantized adder filters have lower computational and memory overheads than A^{2}NNs, increasing their utility in hardware deployment. Although low-bit-width Q-A^{2}NNs exhibit significantly reduced RSSC accuracy compared to A^{2}NNs, this issue can be alleviated by employing the proposed MDD quantization strategy, which combines a weight-debiasing (WD) strategy, which reduces performance degradation due to deviations in the quantized weights, with a feature-debiasing (FD) strategy, which enhances the classification performance of Q-A^{2}NNs through minimizing deviations among the output features of each layer. Extensive experiments and analyses demonstrate that the proposed SSDQ method can efficiently quantize A^{2}NNs to obtain Q-A^{2}NNs with low computational and memory overheads while maintaining comparable performance to A^{2}NNs, thus having high potential for onboard RSSC.

Keywords: onboard processing; deep learning; scene classification; quantization; all-adder neural network

check for updates

Citation: Zhang, N.; Chen, H.; Chen, L.; Wang, J.; Wang, G.; Liu, W. Q-A^{2}NN: Quantized All-Adder Neural Networks for Onboard Remote Sensing Scene Classification. *Remote Sens.* **2024**, *16*, 2403. https://doi.org/10.3390/rs16132403

Academic Editor: Jon Atli Benediktsson

Received: 23 May 2024
Revised: 20 June 2024
Accepted: 26 June 2024
Published: 30 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Remote sensing scene classification (RSSC) is of critical importance in interpreting remote sensing imagery and has been widely employed in disaster detection, urban planning, environmental monitoring, and national security tasks [1]. Deep learning-based methods have recently gained widespread application in the context of RSSC due to their potent feature abstraction and generalization capabilities [2]. In particular, convolutional neural network (CNN)-based RSSC approaches have emerged as a key focus of RSSC research [3,4].

For traditional remote sensing processing, images collected by satellites are downloaded to perform RSSC for interpretation [5,6]. Recently, the volume and resolution of the acquired remote sensing images have significantly increased. Unfortunately, the relatively limited improvement in the data downlink bandwidth imposes considerable transmission pressure [7,8]. Furthermore, the travel time of satellites increases the total latency for users to obtain the processing results [9], posing challenges in the context of time-constrained tasks such as military surveillance, natural disasters, and emergency situations. Thus,

deploying deep learning-based models on satellite edge devices for onboard processing comprises an intuitive solution [10]. Nevertheless, most existing CNN-based RSSC methods, although showcasing remarkable performance, necessitate billions of multiplication and addition operations, along with numerous parameters. Given the constrained computational and memory resources of space platforms, deploying these methods directly on edge devices in space platforms proves challenging [11].

Many researchers have employed model compression methods to reduce model complexity, mainly including low-rank decomposition [12,13], pruning [14,15], knowledge distillation [16], and quantization [17] approaches. Low-rank decomposition methods reduce the redundancies in the specified layers by decomposing the large matrix of convolution kernels; however, these methods cannot perform global parameter compression and are not applicable with the common 1×1 convolution kernel [18]. Pruning methods reduce the computational complexity by removing redundant weights in CNN models; however, they require manual definition of the pruning criteria and fine-tuning to maintain model performance, which is time-consuming and sub-optimal [19,20]. Knowledge distillation methods aim to enable lightweight models to achieve similar performance as larger models through distilling knowledge from the large model to the lightweight model; however, the design of lightweight models remains an additional challenge [21]. Quantization approaches reduce the number of parameters and computational demand through converting floating-point numerical representations of models into integer representations; however, ensuring the accuracy of low-bit-width quantized models poses a significant challenge [22,23]. Moreover, many studies have shown that deploying multiplication operations on edge devices entails more resources and energy overheads, compared to deploying addition operations [24,25]. Figure 1 shows the energy and area costs for various operations in 45 nm ASICs at 0.9 V [24]. Most of the multiplication operations in CNN models are concentrated in convolutional layers [26]. Although the above-mentioned model compression methods can achieve a reduction in model complexity, optimized CNN-based models still heavily rely on convolutions involving billions of multiplication operations. Consequently, directly applying such models for onboard deployment remains challenging.

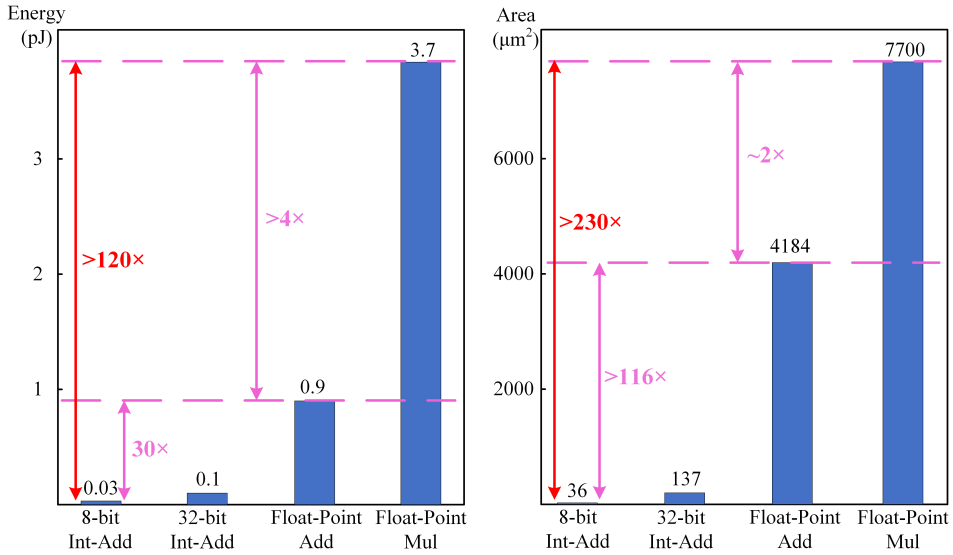


Figure 1. Energy and area overheads for different operations in 45 nm ASICs under 0.9 V.

AdderNet, which is based on adder filters, has recently been proposed [27], allowing for a significant reduction in the number of multiplication operations in the network. However, AdderNet maintains convolutional filters in the first and last layers. Subsequently, Zhang et al. [28] proposed a novel basic neural network, named the all-adder neural network (A^2NN), which converts all multiplications in the convolutional layers into

additions. Nevertheless, despite this breakthrough, A^2NNs have a comparable number of parameters to CNNs. Given the limited memory resources of edge devices on space platforms, quantization approaches can be incorporated to further enhance the hardware efficiency when deploying A^2NN. Compared with other model compression methods, quantization methods can effectively reduce the memory overhead without changing the original network structure. Moreover, the use of integer operations can further reduce the computational overhead during deployment. Thus, the quantized A^2NN (Q-A^2NN) has the advantages of low computational and memory overheads during deployment on edge devices and, thus, is more suitable for onboard RSSC than CNNs and A^2NNs. However, directly applying traditional quantization schemes to quantized A^2NNs re-introduces numerous multiplication operations, which causes forfeiture of their inherent advantage of low computational overhead. Moreover, low-bit-width quantized models usually suffer from significant performance degradation [25,29].

To address these issues, this article proposes a novel shared scaling factor-based debiasing quantization (SSDQ) method to reduce the memory overhead of A^2NNs while minimizing performance degradation. The proposed SSDQ method includes a powers-of-two (POT)-based shared scaling factor quantization scheme and a multi-dimensional de-biasing (MDD) quantization strategy. Specifically, the POT-based shared scaling factor quantization scheme converts the adder filters with 32-bit floating-point (FP32) input activations, weights, and operations to quantized adder filters with hardware-friendly integer input activations, weights, and operations. Thus, the Q-A^2NNs composed of quantized adder filters have lower computational and memory overheads than existing basic networks such as CNNs and A^2NNs. The reduced accuracy of Q-A^2NN for RSSC tasks can be alleviated through the proposed MDD quantization strategy. The MDD strategy combines a weight-debiasing (WD) strategy, which reduces the performance degradation caused by deviations in the quantized weights, with a feature-debiasing (FD) strategy, which enhances the classification performance of Q-A^2NNs through minimizing the deviation in the output features of each layer.

The main contributions of this article can be summarized as follows:

- A novel shared scaling factor-based debiasing quantization method is proposed to reduce the hardware resource overheads of A^2NNs while minimizing performance degradation, which includes a POT-based shared scaling factor quantization scheme and an MDD quantization strategy.
- A POT-based shared scaling factor quantization scheme is proposed to quantize the adder filters in the A^2NN. The proposed quantization scheme converts the input activations and weights of the adder filters from floating-point to integer type, thereby transforming the floating-point addition operations in the adder filters into hardware-friendly integer addition and bit-shift operations.
- An MDD quantization strategy combining the WD and FD strategies is proposed to effectively prevent the decrease in accuracy of Q-A^2NNs due to the deviations in weights and features during quantization. The WD strategy mitigates the performance degradation of Q-A^2NNs by correcting deviations in the quantized weight distribution. It re-defines the weight scaling factor when the weight distribution is skewed and spans considerably beyond the target quantization range, ensuring an adequate quantization range for weights densely distributed near zero. The FD strategy enhances the classification performance of Q-A^2NNs by minimizing deviations among the output features across layers, thus aligning the output features of the intermediate and last layers in the Q-A^2NN with those of the corresponding layers in the A^2NN, reducing quantization errors in a layer-by-layer manner, and improving the feature extraction ability of the Q-A^2NN.

To evaluate the efficacy of the proposed SSDQ method tailored for quantizing A^2NNs, exhaustive experiments are conducted using five commonly used RSSC data sets. The experimental results demonstrate that Q-A^2NNs with low computational and memory

overheads and minimal performance degradation for onboard RSSC can be obtained by employing the proposed SSDQ method.

The remainder of this article is structured as follows: Section 2 provides an overview of the related works. Section 3 covers preliminary knowledge about A^2NNs and traditional quantization schemes for CNNs. Section 4 elaborates on the proposed SSDQ method tailored for quantizing A^2NNs. Section 5 details and analyses the experimental results. Finally, Section 6 concludes the article.

2. Related Works

In this section, we present a brief overview of the related works on RSSC and quantization.

2.1. Remote Sensing Scene Classification

RSSC annotates remote sensing scene images with specific high-level semantic categories, which can be effectively analyzed to obtain meaningful and valuable semantic information.

Existing RSSC methods are primarily classified into three categories: handcrafted feature-based methods [30,31], feature encoding methods [32,33], and deep learning-based methods [34,35]. Recently, deep learning-based methods—particularly CNN-based methods—have made significant strides in RSSC due to their robust feature abstraction and generalization abilities. For instance, Sun et al. [36] integrated hierarchical feature aggregation and interference information elimination schemes into a deep network. The proposed method hierarchically aggregates complementary information and eliminates interference information among different convolutional features, markedly enhancing RSSC accuracy. Wang et al. [37] presented a sphere loss to learn the unique center for each class. Through incorporating a right-angle triangle constraint, the sphere loss aggregates intraclass features while effectively separating the centers of different classes, thereby enhancing RSSC accuracy through intraclass compactness and superior interclass discrimination abilities. Transformer models have recently demonstrated remarkable performance in computer vision tasks and have been utilized for RSSC. For instance, Bazi et al. [38] introduced the vision transformer (ViT) [39] to solve RSSC tasks. Sha et al. [40] presented a multi-instance vision transformer, which enhances the discriminative capability of traditional ViT models by highlighting the feature response of key local regions and simultaneously learning global features.

Although these deep learning-based RSSC methods can achieve remarkable classification performance, directly deploying them on edge devices is challenging due to their high computational complexity and large number of parameters.

2.2. Quantization

Quantization is a highly effective model compression method for reducing the resource overhead of CNNs. In quantized CNNs, the representations of input activations and weights in convolutional layers are converted from FP32 to integer type. Quantization methods are usually divided into two types: post-training quantization (PTQ) and quantization-aware training (QAT) [41–43]. PTQ methods directly convert pre-trained floating-point networks into integer representation networks without re-training. Although PTQ methods are fast and lightweight, they suffer from severe accuracy degradation. QAT methods model the quantization noise source by re-training the quantized CNN for a limited number of iterations, enabling the quantized CNN to find more optimized solutions than with the PTQ [41].

While existing quantization methods have achieved tremendous success with CNNs, directly applying traditional quantization schemes to quantized A^2NNs re-introduces numerous multiplication operations, which causes the forfeiture of their inherent advantage of low computational overhead. Several recent works have investigated quantizing AdderNets, which contain adder filters [27]. Wang et al. [25] proposed a quantization method for AdderNets that integrates input activations and weights, adopting the same scaling factor to quantize them simultaneously. However, when the distributions of the weights

and input activations show significant differences or low-bit-width quantization, directly using the same scaling factor to quantize the adder kernels leads to quantized models with poor performance. Similarly, Zhang et al. [29] directly adopted the weight scaling factor to quantize the weights and input activations of the adder filters; however, this method led to a low-bit-width quantized model with substantially decreased accuracy. This article proposes a novel SSDQ method to develop low-bit-width Q-A^{2}NNs with comparable performance to A^{2}NNs.

3. Preliminary Knowledge

In this section, the A^{2}NN model and the quantization scheme for CNNs are briefly reviewed and analyzed.

3.1. All-Adder Neural Network

The convolution filter used in CNNs can be defined as:

$$O_{t,x,y} = \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} (I_{k,x+p,y+q} \times F_{t,k,p,q}), \tag{1}$$

where $\boldsymbol{I} \in \mathbb{R}^{c_{in} \times h_{in} \times w_{in}}$ and $\boldsymbol{O} \in \mathbb{R}^{c_{out} \times h_{out} \times w_{out}}$ denote the input activation and output feature tensors, respectively, and $\boldsymbol{F} \in \mathbb{R}^{c_{out} \times c_{in} \times d \times d}$ denotes the weight tensor of the filter. The computation of the standard convolution kernel is shown in Figure 2a. The convolution filters use the L2-norm as a similarity measure. In contrast, the adder filters use the L1-norm to indicate the similarity between the filters and input activations to eliminate multiplication operations [27]. The adder filter is defined as:

$$O_{t,x,y} = -\sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} |I_{k,x+p,y+q} - F_{t,k,p,q}|. \tag{2}$$

The computation of the adder kernel is shown in Figure 2b.

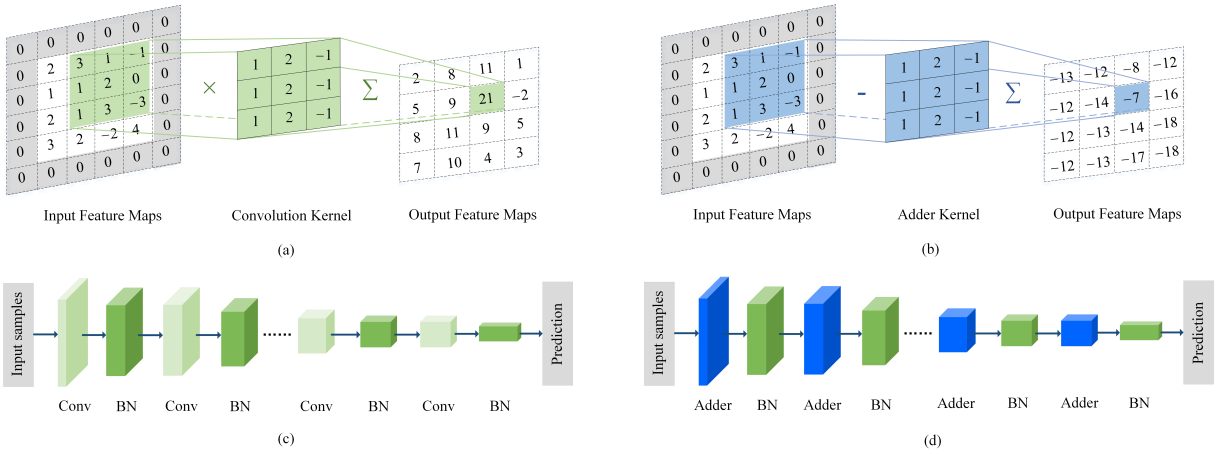


Figure 2. Comparison of the CNN and A^{2}NN structures: (**a**) The convolution kernel used in the CNN; (**b**) the adder kernel used in the A^{2}NN; (**c**) framework of CNN; and (**d**) framework of A^{2}NN.

CNNs employ backpropagation to compute the gradients and stochastic gradient descent for parameter updating. Given that the adder filter does not involve multiplication operations, its partial derivative remains constant. Therefore, the gradients of the adder filter are defined as:

$$\frac{\partial O_{t,x,y}}{\partial F_{t,k,p,q}} = I_{k,x+p,y+q} - F_{t,k,p,q} \tag{3}$$

$$\frac{\partial O_{t,x,y}}{\partial I_{k,x+p,y+q}} = HT\Big(F_{t,k,p,q} - I_{k,x+p,y+q}\Big), \tag{4}$$

where *HT(·)* represents the HardTanh function, defined as:

$$HT(x) = \begin{cases} -1, & x \leq -1 \\ x, & -1 < x < 1 \\ 1, & x \geq 1 \end{cases}. \tag{5}$$

Based on adder filters, Zhang et al. [28] proposed an A^{2}NN, which converts all convolution filters in the CNN into adder filters. The architectures of the CNN and A^{2}NN models are depicted in Figure 2c,d, respectively.

3.2. Quantization Scheme for CNNs

The core concept of quantization is an affine transformation that converts the floating-point operations in the network to efficient integer operations [41]. In this process, a floating-point vector $\boldsymbol{R}$ can be approximated as an integer vector multiplied by a scalar:

$$\boldsymbol{R} = s_R \cdot \boldsymbol{R_{int}}, \tag{6}$$

where s_R is a quantization parameter scaling factor and $\boldsymbol{R_{int}}$ is an integer vector. For N-bit quantization, the elements in $\boldsymbol{R_{int}}$ are N-bit signed integers. In the commonly used hardware-friendly symmetric uniform quantization scheme, the scaling factor s_R is calculated as follows:

$$s_R = \frac{\max(|\max(\boldsymbol{R})|, |\min(\boldsymbol{R})|)}{2^{N-1} - 1}, \tag{7}$$

where max(·) and min(·) are functions utilized to determine the maximum and minimum elements in the given vector, respectively.

In CNNs, the convolution filter in Equation (1) can be re-written in a quantized form by quantizing the input activations and weights:

$$O_{x,y,t} = \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \Big(s_I \cdot I_{int_{k,x+p,y+q}} \times s_F \cdot F_{int_{t,k,p,q}}\Big), \tag{8}$$

where $\boldsymbol{I_{int}} \in \mathbb{R}^{c_{in} \times h_{in} \times w_{in}}$ and $\boldsymbol{F_{int}} \in \mathbb{R}^{c_{out} \times c_{in} \times d \times d}$ represent the quantized input activations and quantized weights, respectively; s_I and s_F represent the quantization parameter scaling factors for the input activations and weights, respectively. According to the distributive and associative laws of multiplication, the quantized convolution filters can use integer-type input activations and weights to perform integer multiplication and addition operations, resulting in quantized output features. Finally, the quantized output features of the quantized convolution filters can be de-quantized with the scaling factors for the input activations s_I and weights s_F. Then, Equation (9) can be re-written as:

$$\begin{aligned} O_{t,x,y} &= s_I s_F \cdot \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \Big(I_{int_{k,x+p,y+q}} \times F_{int_{t,k,p,q}}\Big) \\ &= s_I s_F \cdot O_{int_{t,x,y}}, \end{aligned} \tag{9}$$

where $\boldsymbol{O_{int}} \in \mathbb{R}^{c_{out} \times h_{out} \times w_{out}}$ represents the quantized output feature tensor of the quantized convolution filters.

However, in A^{2}NNs, due to discrepancies in the scaling factors for the input activations s_I and weights s_F, the operation of the adder filter—as shown in Equation (3)—cannot be directly transformed into the form shown in Equation (10). Thus, the Q-A^{2}NN quantized with traditional quantization schemes re-introduces numerous multiplication operations during deployment, leading to forfeiture of the A^{2}NN's inherent advantage of low computational overhead.

4. Method

This section details the SSDQ method tailored for quantizing A^{2}NNs, including a POT-based shared scaling factor quantization scheme and an MDD quantization strategy. The framework of the SSDQ method is presented in Figure 3. Firstly, the POT-based shared scaling factor quantization scheme is devised to quantize the adder filters in the A^{2}NN. This quantization scheme converts adder filters with FP32 input activations, weights, and operations into quantized adder filters with hardware-friendly integer input activations, weights, and operations. Consequently, Q-A^{2}NNs obtained by the POT-based shared scaling factor quantization scheme are entirely composed of quantized adder filters, resulting in lower computational and memory overheads than A^{2}NNs during hardware deployment, although with reduced accuracy for RSSC tasks. Then, the MDD quantization strategy is formulated to mitigate performance degradation of Q-A^{2}NNs due to the deviations in weights and features during the quantization process. The MDD strategy synergistically integrates the WD strategy, which prevents performance degradation caused by deviations in the quantized weights, and the FD strategy, which enhances the classification performance of Q-A^{2}NNs by minimizing the deviations in the output features of each layer. As a result, the Q-A^{2}NNs obtained by the proposed SSDQ method have low computational overhead, low memory overhead, and comparable performance to A^{2}NNs that are more suitable for hardware deployment than CNNs and A^{2}NNs. The principle and details of the POT-based shared scaling factor quantization scheme are expounded in Section 4.1, and the WD strategy and FD strategy in the MDD quantization strategy are detailed in Section 4.2 and Section 4.3, respectively.

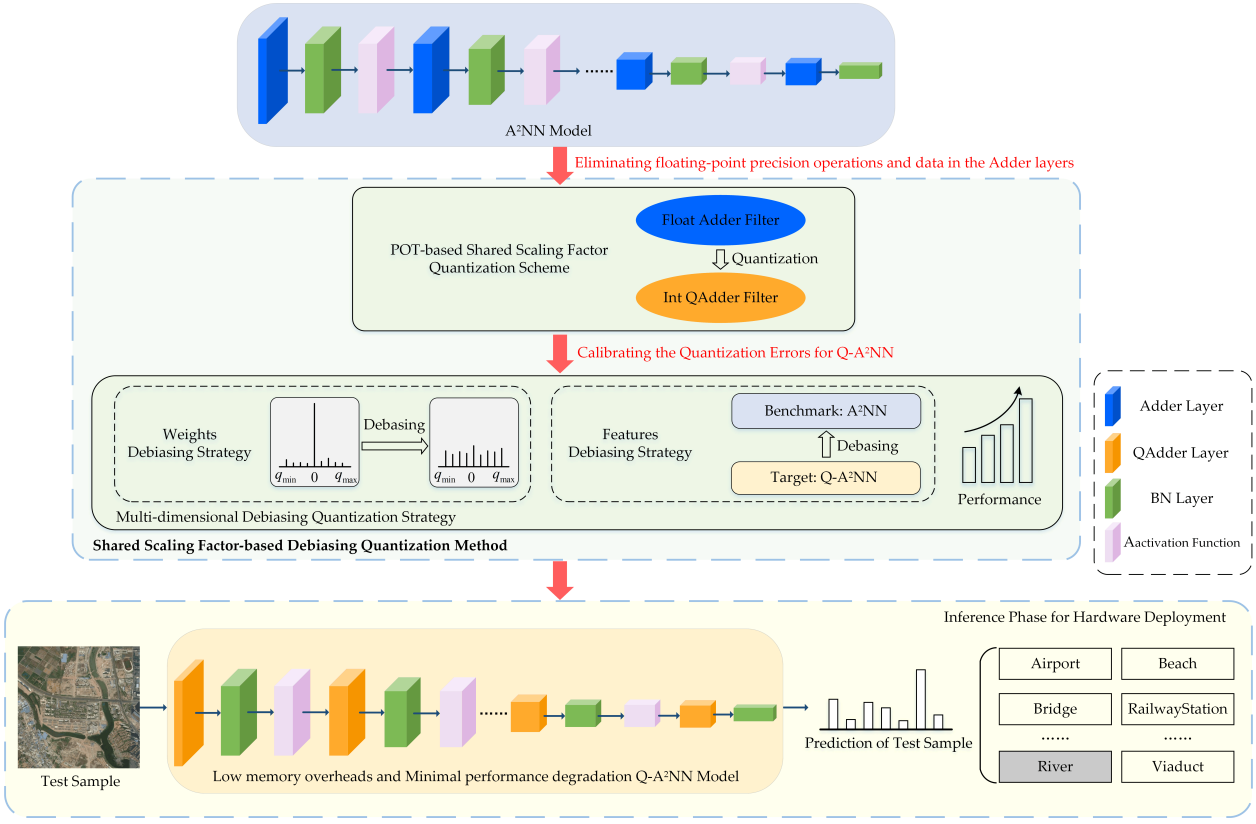


Figure 3. Framework of the proposed SSDQ method tailored for the quantization of A^{2}NNs.

4.1. POT-Based Shared Scaling Factor Quantization Scheme

To address the issue that traditional quantization schemes are not suitable for quantizing A^{2}NNs, a POT-based shared scaling factor quantization scheme is proposed. This quantization scheme can quantize the adder filters in A^{2}NNs, transitioning the input activations, weights, and operations from floating-point to hardware-friendly integer type.

Similar to the quantization of the convolution filter in Equation (9), the adder filter in Equation (3) can be re-written in a quantized form through quantizing the input activations $\boldsymbol{I}$ and weights $\boldsymbol{F}$:

$$O_{t,x,y} = -\sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} |s_I \cdot I_{int_{k,x+p,y+q}} - s_F \cdot F_{int_{t,k,p,q}}|. \tag{10}$$

The scaling factors for the input activations s_I and weights s_F are usually not equal. To avoid introducing multiplication operations into the fully additive neural network during quantization, s_I and s_F can be approximated as POT values, which can be formulated as:

$$s_I = \frac{\max(|\max(\boldsymbol{I})|, |\min(\boldsymbol{I})|)}{2^{N-1}-1} \approx 2^i \tag{11}$$

$$s_F = \frac{\max(|\max(\boldsymbol{F})|, |\min(\boldsymbol{F})|)}{2^{N-1}-1} \approx 2^j, \tag{12}$$

where

$$i = \text{round}\left(\log_2\left(\frac{\max(|\max(\boldsymbol{I})|, |\min(\boldsymbol{I})|)}{2^{N-1}-1}\right)\right) \tag{13}$$

$$j = \text{round}\left(\log_2\left(\frac{\max(|\max(\boldsymbol{F})|, |\min(\boldsymbol{F})|)}{2^{N-1}-1}\right)\right), \tag{14}$$

where round$(\cdot)$ denotes rounding the value to the nearest integer value. According to Equations (12) and (13), the quantized input activations I_{int} and weights F_{int} can be calculated as:

$$I_{int} = \text{clamp}\left(\text{INT}\left(2^{-i}\boldsymbol{I}\right), -2^{N-1}+1, 2^{N-1}-1\right) \tag{15}$$

$$F_{int} = \text{clamp}\left(\text{INT}\left(2^{-j}\boldsymbol{F}\right), -2^{N-1}+1, 2^{N-1}-1\right), \tag{16}$$

where *clamp*$(\cdot)$ confines the quantized elements within the target quantization range $\left[-2^{N-1}+1, 2^{N-1}-1\right]$.

According to the above analysis, Equation (11) can be re-written as:

$$O_{t,x,y} = -\sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \left|2^i I_{int_{k,x+p,y+q}} - 2^j F_{int_{t,k,p,q}}\right|. \tag{17}$$

To further the operation of the adder filter into a hardware-friendly form, similar to Equation (10), the POT-based shared scaling factor s_A can be defined according to the relative magnitudes of the scaling factors for the input activations s_I and weights s_F, as follows:

$$s_A = \min(2^i, 2^j). \tag{18}$$

According to Equations (18) and (19), the operation of the adder filter can be converted into a quantized adder filter with a de-quantized operation, similar to Equation (10). Specifically, the process is divided into the three following cases:

(1) If $i < j$:

$$\begin{aligned} O_{t,x,y} &= -s_A \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \left|I_{int_{k,x+p,y+q}} - 2^{j-i} F_{int_{t,k,p,q}}\right| \\ &= -s_A \cdot O_{int_{t,x,y}}. \end{aligned} \tag{19}$$

(2) If $i = j$:

$$\begin{aligned} O_{t,x,y} &= -s_A \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \left|I_{int_{k,x+p,y+q}} - F_{int_{t,k,p,q}}\right| \\ &= -s_A \cdot O_{int_{t,x,y}}. \end{aligned} \tag{20}$$

(3) If $i > j$:

$$\begin{aligned} O_{t,x,y} &= -s_A \sum_{k=0}^{c_{in}} \sum_{p=0}^{d} \sum_{q=0}^{d} \left| 2^{i-j} I_{int_{k,x+p,y+q}} - F_{int_{t,k,p,q}} \right| \\ &= -s_A \cdot O_{int_{t,x,y}}. \end{aligned} \tag{21}$$

With this quantization scheme, all floating-point addition operations in the adder filter can be converted to integer addition operations. Notably, except for the hardware-friendly bit-shift operations, no additional operations are introduced. The quantized adder filter uses integer-type input activations and weights to perform integer addition operations, resulting in quantized output features. Then, the quantized output features are de-quantized using the POT-based shared scaling factor s_A to obtain the final output features of the adder filter.

4.2. Weight-DeBiasing Strategy

During the quantization process, especially in low-bit-width quantization, discernible deviations emerge between the quantized weight distribution and the original weight distribution. The deviations in the quantized weight distribution cause severe performance degradation of the Q-A^{2}NN model. Consequently, a WD strategy is devised to correct the quantized weight distribution. This subsection first analyzes the causes of the deviations in the weight distribution after quantization. Then, the details of the proposed WD strategy are presented.

The weights of adder filters in well-trained A^{2}NN models often follow Laplacian distributions [27,28]; therefore, for a well-trained A^{2}NN model, many weights are concentrated around zero. Figure 4a illustrates the parameter count histogram for the adder filter in the first layer of the well-trained A^{2}NN-VGGNet-11 model [28]. It can be seen that the weight distribution of the adder filters is skewed. While the weight distribution of this filter ranges from -50 to 40, most weights are distributed between -2 and 2, accounting for over 80%. In contrast, the weights with values below -30 or above 20 are rare, accounting for less than 1%. As the weight scaling factor s_F is determined according to the boundary values of the weight distribution F_{min} and F_{max}, the quantized weights may seriously deviate in low-bit-width Q-A^{2}NN. As shown in Figure 4b, when the target quantization range is much smaller than the actual weight distribution range, many weights that are densely distributed around zero are compressed to the integer zero after quantization. Thus, most of the information in the well-trained full-precision A^{2}NN model is lost during quantization, resulting in a severe decrease in accuracy. Notably, the deviation in the weight distribution after quantization escalates as the quantization bit width decreases.

The proposed WD strategy can effectively alleviate the severe decrease in accuracy of the Q-A^{2}NN caused by deviations in the weight distribution during quantization, the details of which are presented in Figure 4c. When the weight distribution is skewed and the weight distribution range is much more extensive than the target quantization range, the median value F_{Me} reflects the actual weight distribution better than F_{min} and F_{max}. Therefore, when the weight scaling factor s_F obtained using the boundary values F_{min} and F_{max} exceeds 1, the weight scaling factor s_F is re-defined as:

$$s_F = \frac{\max\left|(F'_{Me}|, |F''_{Me}|\right)}{2^{N-2}-1} \approx 2^j, \tag{22}$$

where

$$j = \text{round}\left(\log_2\left(\frac{\max\left|(F'_{Me}|, |F''_{Me}|\right)}{2^{N-2}-1}\right)\right), \tag{23}$$

where F'_{Me} and F''_{Me} represent the negative median value and positive median value in the weight distribution, respectively.

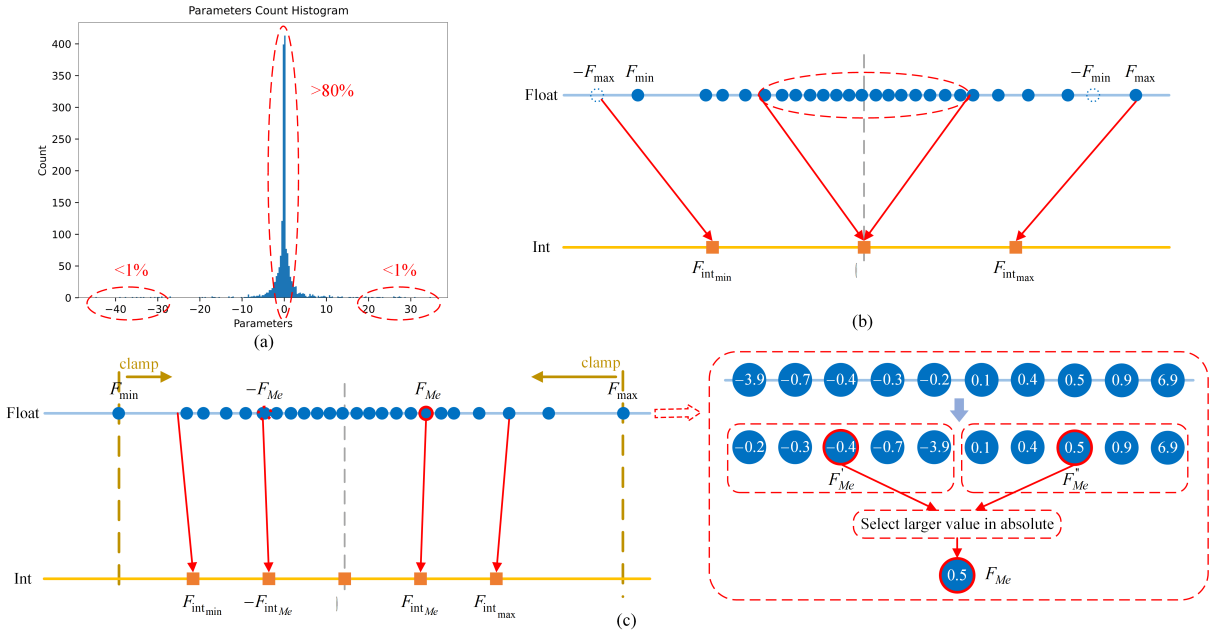


Figure 4. An overview of the proposed WD strategy: (**a**) Parameter count histogram for the adder filter in the first layer of the well-trained A^{2}NN-VGGNet-11 model; (**b**) the reason for the deviation in the quantized weight distribution; and (**c**) demonstration of the WD strategy.

As shown in Figure 4c, the median value of the weight distribution F_{Me} can be mapped to the median value of the quantization range $F_{\text{int}_{Me}}$ using the re-defined weight scaling factor s_F. The WD strategy ensures an adequate quantization range for the weights densely distributed near zero in the adder filters, effectively alleviating the performance degradation resulting from substantial information loss during the quantization process. In particular, the proposed WD strategy does not introduce additional computational or memory overhead during the inference phase of the Q-A^{2}NN model.

4.3. Feature-DeBiasing Strategy

During the quantization process of A^{2}NNs, not only does the weight distribution present discernible deviations, but the output features of each layer also have discernible deviations due to the accumulation of quantization errors. Consequently, another debiasing strategy integrated into the multi-dimensional de-biasing strategy—namely, the FD strategy—is devised to improve the classification performance of the Q-A^{2}NN by reducing the deviation among the output features of each layer. The framework of the FD strategy is interpreted in Figure 5. The well-trained A^{2}NN is employed as the benchmark model, and the Q-A^{2}NN is set as the target model. By aligning the output features of each intermediate layer in the Q-A^{2}NN model with the corresponding layer's output features in the A^{2}NN model, the quantization errors of the middle layers can be effectively reduced in a layer-by-layer manner. With this strategy, the accumulation of quantization errors—which considerably affect the subsequent layers in the Q-A^{2}NN model—can be mitigated. Furthermore, through aligning the output features of the last layer in the target Q-A^{2}NN model with the output features of the last layer in the benchmark A^{2}NN model, the feature extraction ability of the target Q-A^{2}NN model can be brought as close as possible to that of the benchmark A^{2}NN model.

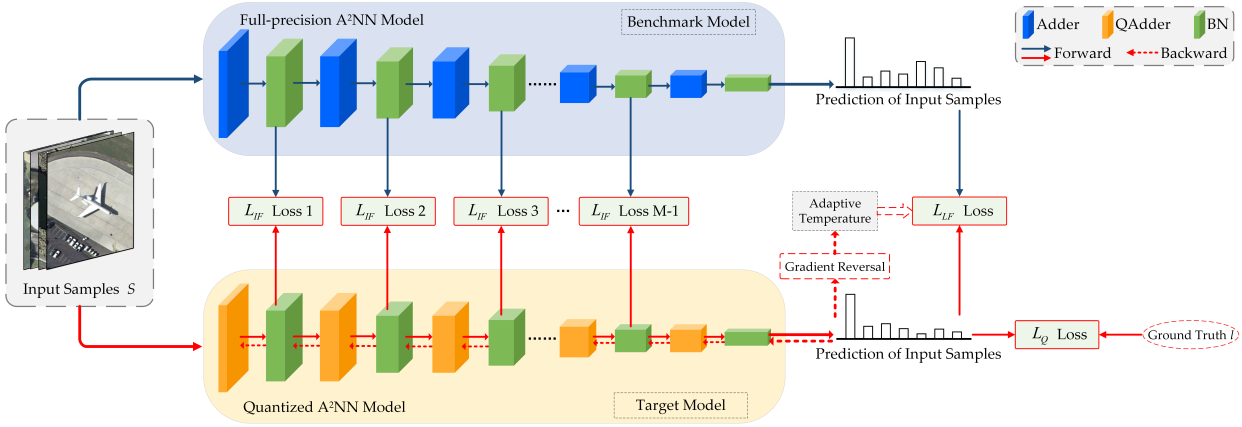


Figure 5. Framework of the proposed FD strategy. The well-trained A^{2}NN model is employed as the benchmark model to enhance the classification performance of the target Q-A^{2}NN model by reducing the deviation among the output features of each layer.

Thus, a joint loss function L_{FD} of the FD strategy is designed to optimize the Q-A^{2}NN model, which can be defined as follows:

$$L_{FD} = L_Q + \lambda(L_{IF} + L_{LF}), \tag{24}$$

where L_Q represents the quantization loss, L_{IF} represents the feature loss of the intermediate layers, and L_{LF} represents the feature loss of the final layer. These losses are used to construct the joint loss function L_{FA}, and λ is a hyperparameter for balancing these loss functions.

The quantization loss function L_Q is used to minimize the distance between the ground truth (GT) and classification logits of the Q-A^{2}NN model, which can be expressed as follows:

$$L_Q = CE[f_{Q-A^2NN}(S), l], \tag{25}$$

where $CE(\cdot)$ represents the cross-entropy loss function, $f_{Q-A^2NN}(\cdot)$ denotes the function set of the Q-A^{2}NN model, and l denotes the GT of the training samples.

The intermediate feature loss function L_{IF} is devised to measure the gap between the output features of the intermediate batch normalization (BN) layers in the Q-A^{2}NN and benchmark A^{2}NN models. The A^{2}NN model employs a BN layer after each adder layer to stabilize the feature distribution. Thus, the deviation among the output features of each intermediate layer due to quantization errors can be reduced as much as possible through minimizing L_{IF}, thereby attenuating the adverse effects of accumulated quantization errors on subsequent layers. The intermediate feature loss function L_{IF} is formulated as follows:

$$L_{IF} = \sum_{b=1}^{M-1} MSE(\hat{F}_b, F_b), \tag{26}$$

where M denotes the number of BN layers; $\hat{F}_b$ and F_b are the output features of the bth BN layer in the Q-A^{2}NN and benchmark A^{2}NN models, respectively; and $MSE(\cdot)$ represents the mean squared error operator, expressed as follows:

$$MSE(\hat{F}_b, F_b) = \frac{1}{B}\sum_{i=0}^{B} \| \hat{F}_b - F_b \|_2^2, \tag{27}$$

where B denotes the batch size in the training phase.

The last feature loss function L_{LF} is formulated to determine the disparity between the class probabilities in the last layer of the Q-A^{2}NN and A^{2}NN models. Due to information loss during the quantization process, the feature extraction ability of the Q-A^{2}NN model is

worse than that of the A^{2}NN model. L_{LF} is minimized to optimize the Q-A^{2}NN model by ensuring that the classification boundary of the last layer in the Q-A^{2}NN model imitates that of the benchmark A^{2}NN model. The Kullback–Leibler (KL) divergence is used to construct the loss function L_{LF}, which is formulated as follows:

$$\begin{aligned} L_{LF} &= T^2 \cdot KLDiv[P_{Q-A^2NN}, P_{A^2NN}] \\ &= T^2 \cdot \sum_{k=0}^{C} P_{Q-A^2NN}(k) \log \frac{P_{Q-A^2NN}(k)}{P_{A^2NN}(k)}, \end{aligned} \tag{28}$$

where C represents the number of classes represented by the training samples, T is a crucial hyperparameter dictating the softness in the class probability, and P_{Q-A^2NN} and P_{A^2NN} denote the class probability for the target Q-A^{2}NN model and the benchmark A^{2}NN model, respectively. The pth elements in P_{Q-A^2NN} and P_{A^2NN} can be computed as:

$$P_{Q-A^2NN}(p) = \log \frac{e^{\frac{Q-A^2NN(p)}{T}}}{\sum_{c=0}^{C} e^{\frac{Q-A^2NN(c)}{T}}} \tag{29}$$

$$P_{A^2NN}(p) = \frac{e^{\frac{A^2NN(p)}{T}}}{\sum_{c=0}^{C} e^{\frac{A^2NN(c)}{T}}}. \tag{30}$$

Notably, determining the appropriate hyperparameter T for Q-A^{2}NN models with different bit widths is difficult and resource-intensive. Inspired by [44], an adaptive temperature is introduced for adaptation to Q-A^{2}NN models with different bit widths. Furthermore, the adaptive temperature is learned utilizing a reverse gradient strategy that aims to maximize L_{LF}. Through gradually increasing the challenge posed to the Q-A^{2}NN model in emulating the classification boundary of the well-trained full-precision A^{2}NN model, an adversarial process is introduced to continually improve the Q-A^{2}NN model's feature extraction capability. Consequently, the feature extraction ability of the Q-A^{2}NN model approaches that of the benchmark A^{2}NN model. The adaptive temperature T can be updated using the reversed gradient as follows:

$$T^{p+1} = T^p - lr_T \times (-\frac{\partial L_{FD}}{\partial T^p}), \tag{31}$$

where lr_T denotes the learning rate of the adaptive temperature, and T^p and T^{p+1} denote the value of the adaptive temperature in the pth and $(p+1)$th iterations, respectively.

Through the above analysis, it can be concluded that Q-A^{2}NN models with low computational overhead, low memory overhead, and minimal performance degradation can be obtained using the proposed SSDQ method tailored for the quantization of A^{2}NNs.

5. Experiments

To assess the efficacy of the proposed SSDQ method tailored for quantizing A^{2}NNs, extensive experiments were conducted using five commonly used RSSC data sets. These experiments were accelerated by an NVIDIA Titan RTX graphical processing unit and performed using PyTorch 1.8 [45].

5.1. Data Set Description and Pre-Processing

Five RSSC data sets were utilized to conduct the experiments: WHU-RS19 (WHU) [46], UC-Merced Land Use Data Set (UCM) [47], SIRI-WHU [33], RSSCN7 [48], and Aerial Image Data Set (AID) [49]. These five widely employed data sets encompass numerous scene classes, boasting high inter-class similarity and within-class diversity. A portion of images from each data set was randomly chosen as the training samples, while the remainder

served as the testing samples. The essential information and settings of these five data sets for the following experiments are detailed in Table 1.

Table 1. Information and settings of five RSSC data sets used in the experiments.

Data Set	WHU	UCM	SIRI-WHU	RSSCN7	AID
Classes	19	21	12	7	30
Total images	1005	2100	2400	2800	10,000
Images per class	∼50	100	200	400	220∼420
Training sample ratio	0.8	0.8	0.4	0.2	0.2
Testing sample ratio	0.2	0.2	0.6	0.8	0.8
Resolution (m)	up to 0.5	0.3	2	-	0.5∼8
Image size	600 × 600	256 × 256	200 × 200	400 × 400	600 × 600
Data source	Google Earth	USGS	Google Earth	Google Earth	Google Earth

Moreover, some data augmentation methods were applied to the samples in each data set, following recent research [28,50]. Firstly, all samples used in the following experiments were resized to 256 × 256 pixels. Then, the training samples and testing samples were random-cropped and center-cropped to 224 × 224 patches. Furthermore, the cropped training samples were flipped horizontally.

5.2. Evaluation Metrics

In accordance with recent research in the RSSC field [28,51], we adopted six widely accepted metrics to evaluate the classification performance of the proposed method: overall accuracy (*OA*), confusion matrix, precision (*Pre*), recall (*Rec*) and *F*1 score (*F*1). Moreover, two widely used indicators were adopted to evaluate the computational and memory overheads of different models: the number of operations (OPs) and the model size.

The *OA* reflects the overall classification performance, which is defined as follows:

$$OA = \frac{N_c}{N_T}, \tag{32}$$

where N_c denotes the number of correctly classified test samples, while N_T denotes the number of test samples.

The confusion matrix illustrates the detailed confusion degree and between-class classification errors of the model [28]. The entry in the nth column and mth row in the confusion matrix can be calculated as:

$$C_{m,n} = \frac{N_{m,n}}{N_m}, \tag{33}$$

where $N_{m,n}$ represents the number of samples of the mth class classified as the nth class, and N_m represents the total number of images in the mth class.

The *Pre* reflects the proportion of true positive samples among all samples predicted as positive. The *Pre* of the ith class is defined as follows:

$$Pre_i = \frac{TP_i}{TP_i + FP_i}, \tag{34}$$

where TP and FP represent the number of true positives and false positives, respectively.

The *Rec* reflects the proportion of true positive samples among all truly positive samples in the ground truth. The *Rec* of the ith class is defined as follows:

$$Rec_i = \frac{TP_i}{TP_i + FN_i}, \tag{35}$$

where TP and FN represent the number of true positives and false negatives, respectively.

The $F1$ balances Pre and Rec by calculating the harmonic mean of Pre and Rec, which can better reflect the generalization ability of the model. The $F1$ of the ith class is defined as follows:

$$F1_i = \frac{2 \times Pre_i \times Rec_i}{Pre_i + Rec_i}. \tag{36}$$

Notably, the macro-average method was used to obtain the performance metrics of Pre, Rec, and $F1$ across the entire data set by directly averaging the evaluation metrics of different categories.

5.3. Experimental Settings

5.3.1. Network

In our experiments, we employed two representative neural network architectures as network backbones: ResNet-18 and VGGNet-11. To avoid redundancy, only the final fully connected (FC) layer was retained as the classifier in VGGNet-11. To facilitate deployment, the FC layer in both backbones was replaced with a 1×1 convolutional layer. Since the output of the adder filter is always negative, the BN layer was introduced to normalize the output of the adder layers to an appropriate range. BN layers were added after each convolutional layer and FC layer in both backbones to maintain consistency across structures. Additionally, the pooling layers and activation functions in the network backbones were retained to enhance the robustness and feature extraction capabilities of the network model.

Six models (sets) were developed based on each backbone. For the ResNet-18 backbone, the following models (sets) were created: the CNN-ResNet-18 model, binary neural network (BNN)-ResNet-18 model, AdderNet-ResNet-18 model, A^{2}NN-ResNet-18 model, quantized CNN (Q-CNN)-ResNet-18 model set, and Q-A^{2}NN-ResNet-18 model set. Similarly, for the VGGNet-11 backbone, the CNN-VGGNet-11 model, BNN-VGGNet-11 model, AdderNet-VGGNet-11 model, A^{2}NN-VGGNet-11 model, Q-CNN-VGGNet-11 model set, and Q-A^{2}NN-VGGNet-11 model set were created. The BNN [52] is a lightweight model utilizing only two possible values to constrain weights. The Q-CNN and Q-A^{2}NN model sets encompass multiple models with varying quantization bit widths, with the quantization bit width N taking values from $\{4, 5, 6, 7, 8, 10\}$. The models with different quantization bit widths in the Q-CNN-ResNet-18 model set are denoted as Q-CNN-ResNet-18-Nbit models, while the models with different quantization bit widths in the Q-A^{2}NN-ResNet-18 model set are denoted as Q-A^{2}NN-ResNet-18-Nbit models. The models with different quantization bit widths in the Q-CNN-VGGNet-11 and Q-A^{2}NN-VGGNet-11 model sets are denoted as Q-CNN-VGGNet-11-Nbit models and Q-A^{2}NN-VGGNet-11-Nbit models, respectively. For the Q-CNN and Q-A^{2}NN model sets, the input activations and weights of all convolutional or adder layers were quantized. To ensure fairness, none of the full-precision models were loaded with pre-trained parameters.

5.3.2. Hyperparameter Settings

Following [28], all full-precision models were trained for 300 epochs. They were trained using stochastic gradient descent (SGD) with a batch size of 32, momentum of 0.9, and weight decay of 0.0005. The learning rate had an initial value of 0.05, which changed according to the cosine learning rate decay [53]. Similarly, all quantized models were trained using the quantization-aware training (QAT) method for 60 epochs. They were trained using SGD with batch size of 64, momentum of 0.9, and weight decay of 0.0005. The learning rate had an initial value of 0.002, and the cosine learning rate decay was employed.

5.4. Hyperparameter Analysis

We conducted hyperparameter sensitivity experiments on the RSSCN7 data set using four Q-A^{2}NN models: Q-A^{2}NN-ResNet-18-4bit, Q-A^{2}NN-ResNet-18-8bit, Q-A^{2}NN-VGGNet-11-4bit, and Q-A^{2}NN-VGGNet-11-8bit. These experiments aimed to analyze the influence of the balance coefficient hyperparameter λ in the joint loss function L_{FD} on the

performance, in order to select the optimal value of this hyperparameter. Subsequently, this selected value was applied in the following experiments on the other four data sets with the remaining Q-A^{2}NN models.

The λ value was varied within the set $\{0.1, 0.5, 1, 3, 5, 7, 10, 12, 15\}$. The variation in the OA with λ is depicted in Figure 6. The proposed SSDQ method demonstrated satisfactory OA across a wide range of λ values, with the best classification results of the four models achieved when λ was set to 1 within the given set. Therefore, in the following experiments, λ was set to 1.

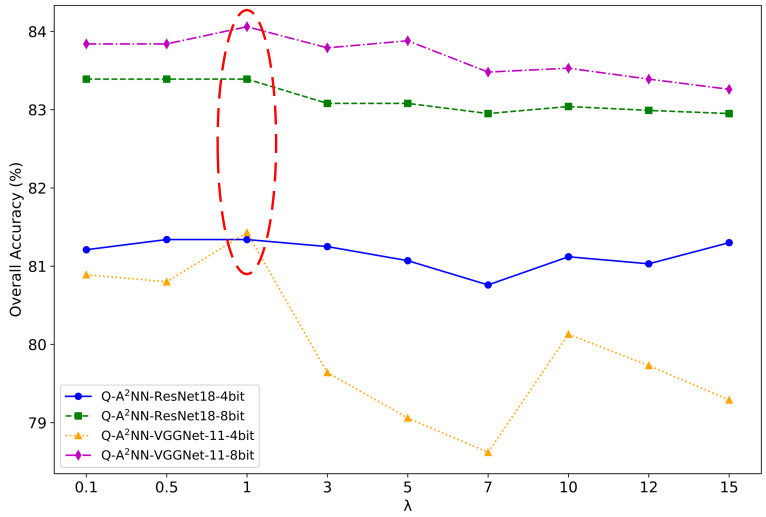


Figure 6. *OA* of four Q-A^{2}NN models with different hyperparameter values on the RSSCN7 data set.

5.5. Comparison with Other Approaches

To demonstrate the effectiveness and advancement of the proposed SSDQ method tailored for the quantization of A^{2}NNs, extensive experiments were conducted on the five commonly used RSSC data sets. Table 2 presents the classification results of the different models on the five data sets. To validate the reliability of the classification results, each experiment was repeated three times. Moreover, to compare the computational and memory overheads during the inference phase, the OPs and Params of the aforementioned models were determined, which are presented in Table 3. The major computational layers in the table refer to convolutional/adder/quantized convolutional/quantized adder/binarize layers, while the other layers primarily consist of activation and BN layers. As the computational and memory overheads of the other layers are substantially lower than those of the major computational layers, they can be disregarded. Thus, the computational and memory overheads of models were primarily influenced by the convolutional/adder/quantized convolutional/quantized adder/binarized layers.

As presented in Tables 2 and 3, the Q-A^{2}NN model sets (POT), obtained using only the proposed POT-based shared scaling factor quantization scheme, had lower memory overhead than the A^{2}NN models and showed great classification performance with 8-bit quantization. However, as the quantization bit-width decreased, the performance of the Q-A^{2}NN model sets (POT) decreased sharply. For example, on the RSSCN7 data set, the Q-A^{2}NN-ResNet-18-8bit model (POT) achieved 83.20% accuracy, which is 0.3% higher than that of the A^{2}NN model, and the memory overhead was only $\frac{1}{4}$ that of the A^{2}NN model; in contrast, the Q-A^{2}NN-ResNet-18-4bit model (POT) achieved only 20.64% accuracy.

As shown in Table 2, for most RSSC data sets, the Q-A^{2}NN model sets (SSDQ) obtained with the proposed SSDQ method achieved comparable classification accuracies to the A^{2}NN models. For some data sets, the accuracies of the Q-A^{2}NN model sets (SSDQ) even surpassed those of the A^{2}NN models. For instance, the OA of the Q-A^{2}NN-ResNet-18-6bit model (SSDQ) was improved by approximately 0.28%, 0.42%, and 0.98% on the RSSCN7, AID, and WHU data sets, respectively. The Q-A^{2}NN-ResNet-18-4bit model

(SSDQ) achieved 90.73% accuracy on the WHU data set, which is the same accuracy as the A^{2}NN model. Moreover, the computational and memory overheads of the Q-A^{2}NN model sets (POT) and Q-A^{2}NN model sets (SSDQ) during the inference phase were identical, as shown in Table 3.

According to Tables 2 and 3, the Q-A^{2}NN model sets (SSDQ) had the same or even lower performance degradation, when compared with the Q-CNN model sets quantized using the symmetric uniform quantization scheme [54,55]. Moreover, the memory overhead of the Q-A^{2}NN model sets (SSDQ) and Q-CNN model sets during the inference phase were identical. For example, the performance degradation of the Q-A^{2}NN-VGGNet-11-4bit models (SSDQ) was nearly 0.19%, 0.49%, and 1.28% lower than that of the Q-CNN-VGGNet-11-4bit models for the RSSCN7, WHU, and SIRI-WHU data sets, respectively. The memory overhead required to deploy these models was the same—approximately 4.42 MB. Notably, on some data sets, the accuracies of Q-A^{2}NN model sets (SSDQ) even surpassed that of full-precision CNN models, and the computational and memory overheads were much lower than those of the CNN models.

Table 2. Classification accuracies of various models on the five RSSC data sets [OA ± STD(%)].

Data Set	Backbone	Precision	Basic Network						
			CNN [28]	Q-CNN [54,55]	BNN [52]	AdderNet [27]	A^{2}NN [28]	Q-A^{2}NN (POT) (Ours)	Q-A^{2}NN (SSDQ) (Ours)
UCM	ResNet-18	Floating-point/Binarize *	96.00 ± 0.66	95.95	50.67 ± 0.92	94.14 ± 0.93	95.62 ± 0.27	95.71	95.71
		8-bit	-	95.79 ± 0.14	-	-	-	94.29 ± 0.24	95.40 ± 0.14
		6-bit	-	95.56 ± 0.14	-	-	-	88.02 ± 0.27	95.40 ± 0.14
		4-bit	-	95.32 ± 0.14	-	-	-	11.98 ± 1.79	95.16 ± 0.14
	VGGNet-11	Floating-point/Binarize	96.10 ± 0.89	96.67	89.09 ± 0.57	94.67 ± 0.46	96.76 ± 0.55	96.9	96.9
		8-bit	-	97.14 ± 0	-	-	-	95.48 ± 0	97.14 ± 0.24
		6-bit	-	97.06 ± 0.14	-	-	-	67.54 ± 0.60	96.51 ± 0.14
		4-bit	-	94.68 ± 0.50	-	-	-	10.16 ± 0.90	94.84 ± 0.28
WHU	ResNet-18	Floating-point/Binarize	92.58 ± 0.87	91.22	60.29 ± 1.12	87.22 ± 0.63	90.24 ± 0.49	90.73	90.73
		8-bit	-	90.73 ± 0	-	-	-	90.40 ± 0.28	92.20 ± 0
		6-bit	-	90.24 ± 0	-	-	-	88.45 ± 0.28	91.71 ± 0
		4-bit	-	89.60 ± 0.29	-	-	-	10.73 ± 0	90.73 ± 0
	VGGNet-11	Floating-point/Binarize	91.90 ± 0.74	91.71	82.73 ± 1.56	89.46 ± 0.74	92.30 ± 0.80	92.2	92.2
		8-bit	-	90.57 ± 0.28	-	-	-	90.89 ± 0.28	91.71 ± 0
		6-bit	-	90.24 ± 0	-	-	-	80.49 ± 0.49	91.22 ± 0
		4-bit	-	90.24 ± 0.97	-	-	-	9.76 ± 0	91.22 ± 0.49
RSSCN7	ResNet-18	Floating-point/Binarize	84.16 ± 0.60	83.71	62.70 ± 0.49	79.98 ± 0.98	82.42 ± 0.60	82.9	82.9
		8-bit	-	83.52 ± 0.63	-	-	-	83.20 ± 0.09	83.36 ± 0.05
		6-bit	-	83.54 ± 0.09	-	-	-	70.61 ± 0.14	83.18 ± 0.11
		4-bit	-	81.85 ± 0.16	-	-	-	20.64 ± 0.20	81.40 ± 0.18
	VGGNet-11	Floating-point/Binarize	82.12 ± 0.42	82.19	78.03 ± 0.62	79.98 ± 0.82	83.29 ± 0.45	83.08	83.08
		8-bit	-	82.78 ± 0.07	-	-	-	83.62 ± 0.28	83.91 ± 0.14
		6-bit	-	82.66 ± 0.05	-	-	-	54.40 ± 0.49	83.96 ± 0.09
		4-bit	-	80.01 ± 0.29	-	-	-	17.49 ± 0	81.09 ± 0.30
AID	ResNet-18	Floating-point/Binarize	85.29 ± 0.56	84.74	42.67 ± 0.22	77.58 ± 0.63	79.15 ± 0.32	78.85	78.85
		8-bit	-	84.75 ± 0.03	-	-	-	79.33 ± 0.03	79.55 ± 0.10
		6-bit	-	84.75 ± 0.03	-	-	-	69.47 ± 0.54	79.27 ± 0.13
		4-bit	-	83.07 ± 0.03	-	-	-	5.64 ± 1.65	76.60 ± 0.05
	VGGNet-11	Floating-point/Binarize	83.28 ± 0.59	83.06	66.65 ± 0.85	81.28 ± 0.65	83.46 ± 0.23	83.36	83.36
		8-bit	-	83.30 ± 0.08	-	-	-	84.49 ± 0.01	83.92 ± 0.07
		6-bit	-	83.13 ± 0.08	-	-	-	72.00 ± 0.03	83.87 ± 0.11
		4-bit	-	80.29 ± 0.07	-	-	-	5.14 ± 0.51	78.06 ± 0.09
SIRI-WHU	ResNet-18	Floating-point/Binarize	91.90 ± 0.24	91.94	57.07 ± 1.45	86.22 ± 0.91	89.61 ± 0.24	89.86	89.86
		8-bit	-	91.74 ± 0.07	-	-	-	89.65 ± 0.07	89.81 ± 0.08
		6-bit	-	91.58 ± 0.04	-	-	-	76.78 ± 0.14	89.05 ± 0.04
		4-bit	-	89.74 ± 0.15	-	-	-	16.41 ± 0.87	87.59 ± 0.23
	VGGNet-11	Floating-point/Binarize	87.53 ± 0.40	87.64	80.56 ± 0.21	86.68 ± 0.73	88.80 ± 0.61	88.33	88.33
		8-bit	-	87.78 ± 0	-	-	-	89.17 ± 0.07	89.05 ± 0.23
		6-bit	-	87.45 ± 0.11	-	-	-	31.69 ± 1.51	89.40 ± 0.35
		4-bit	-	83.89 ± 0.24	-	-	-	11.39 ± 0.84	85.86 ± 0.21

* For the CNN, AdderNet, and A^{2}NN models, this precision refers to floating-point type; for the BNN model, this precision refers to binary type.

The classification accuracy of the Q-A^{2}NN model sets (SSDQ) surpassed that of the AdderNet models on most of the data sets, according to Table 2. For example, the Q-A^{2}NN-ResNet-18-4bit models (SSDQ) enhanced the OA compared to the AdderNet models by approximately 1.42%, 1.37%, 3.51%, and 1.02% on the RSSCN7, SIRI-WHU, WHU, and

UCM data sets, respectively; the Q-A^{2}NN-VGGNet-11-4bit models (SSDQ) enhanced the OA by approximately 1.76%, 1.11%, and 0.17% on the WHU, RSSCN7, and UCM data sets, respectively. Furthermore, the computational and memory overheads of the Q-A^{2}NN model sets (SSDQ) were lower than those of the AdderNet models, as presented in Table 3.

As illustrated in Tables 2 and 3, while the BNN models exhibited smaller computational and memory overheads, compared to other models, their accuracy was significantly lower. For instance, the accuracy of the BNN-ResNet-18 model was only 60.29% on the WHU data set, while the Q-A^{2}NN-ResNet-18-4bit model (SSDQ) achieved an accuracy of 90.73% on the same data set. As such, the reduced accuracy of BNN models can be considered unacceptable.

Table 3. Computational and memory overheads of different models.

Backbone	Basic Network	Computational Overhead					Memory Size *		
		Major Computational Layers			Other Layers	OPs	Params	Major Computational Layers	Other Layers
		Add	Mul	XNOR	MACs				
ResNet-18	CNN [28]	1.81 G	1.81 G	0	4.98 M	3.63 G	42.79 MB	42.66 MB	0.04 MB
	AdderNet [27]	3.56 G	59.01 M	0	4.98 M	3.63 G	42.79 MB	42.66 MB	0.04 MB
	A^{2}NN [28]	3.62 G	0	0	4.98 M	3.63 G	42.79 MB	42.66 MB	0.04 MB
	BNN [52]	1.81 G	0	1.81 G	4.98 M	3.63 G	1.37 MB	1.33 MB	0.04 MB
	Q-CNN-8bit [54,55]	1.81 G	1.81 G	0	4.98 M	3.63 G	10.70 MB	10.66 MB	0.04 MB
	Q-CNN-6bit [54,55]	1.81 G	1.81 G	0	4.98 M	3.63 G	8.04 MB	8.00 MB	0.04 MB
	Q-CNN-4bit [54,55]	1.81 G	1.81 G	0	4.98 M	3.63 G	5.37 MB	5.33 MB	0.04 MB
	Q-A^{2}NN-8bit (POT) (ours)	3.62 G	0	0	4.98 M	3.63 G	10.70 MB	10.66 MB	0.04 MB
	Q-A^{2}NN-6bit (POT) (ours)	3.62 G	0	0	4.98 M	3.63 G	8.04 MB	8.00 MB	0.04 MB
	Q-A^{2}NN-4bit (POT) (ours)	3.62 G	0	0	4.98 M	3.63 G	5.37 MB	5.33 MB	0.04 MB
	Q-A^{2}NN-8bit (SSDQ) (ours)	3.62 G	0	0	4.98 M	3.63 G	10.70 MB	10.66 MB	0.04 MB
	Q-A^{2}NN-6bit (SSDQ) (ours)	3.62 G	0	0	4.98 M	3.63 G	8.04 MB	8.00 MB	0.04 MB
	Q-A^{2}NN-4bit (SSDQ) (ours)	3.62 G	0	0	4.98 M	3.63 G	5.37 MB	5.33 MB	0.04 MB
VGGNet-11	CNN [28]	7.49 G	7.49 G	0	14.85 M	15.00 G	35.24 MB	35.22 MB	0.02 MB
	AdderNet [27]	14.93 G	43.36 M	0	14.85 M	15.00 G	35.24 MB	35.22 MB	0.02 MB
	A^{2}NN [28]	14.97 G	0	0	14.85 M	15.00 G	35.24 MB	35.22 MB	0.02 MB
	BNN [52]	7.49 G	0	7.49 G	14.85 M	15.00 G	1.12 MB	1.10 MB	0.02 MB
	Q-CNN-8bit [54,55]	7.49 G	7.49 G	0	14.85 M	15.00 G	8.83 MB	8.81 MB	0.02 MB
	Q-CNN-6bit [54,55]	7.49 G	7.49 G	0	14.85 M	15.00 G	6.62 MB	6.60 MB	0.02 MB
	Q-CNN-4bit [54,55]	7.49 G	7.49 G	0	14.85 M	15.00 G	4.42 MB	4.40 MB	0.02 MB
	Q-A^{2}NN-8bit (POT) (ours)	14.97 G	0	0	14.85 M	15.00 G	8.83 MB	8.81 MB	0.02 MB
	Q-A^{2}NN-6bit (POT) (ours)	14.97 G	0	0	14.85 M	15.00 G	6.62 MB	6.60 MB	0.02 MB
	Q-A^{2}NN-4bit (POT) (ours)	14.97 G	0	0	14.85 M	15.00 G	4.42 MB	4.40 MB	0.02 MB
	Q-A^{2}NN-8bit (SSDQ) (ours)	14.97 G	0	0	14.85 M	15.00 G	8.83 MB	8.81 MB	0.02 MB
	Q-A^{2}NN-6bit (SSDQ) (ours)	14.97 G	0	0	14.85 M	15.00 G	6.62 MB	6.60 MB	0.02 MB
	Q-A^{2}NN-4bit (SSDQ) (ours)	14.97 G	0	0	14.85 M	15.00 G	4.42 MB	4.40 MB	0.02 MB

* The memory size varies slightly for different data sets; here, the AID data set is used as an example.

To evaluate the proposed SSDQ method from multiple dimensions, Table 4 presents a comparison of the Pre, Rec, and F1 metrics for CNN, A^{2}NN, Q-CNN, Q-A^{2}NN (POT), and Q-A^{2}NN (SSDQ) models based on different backbones on the WHU data set. From the numerical comparisons in Table 4, it can be seen that the Q-A^{2}NN (SSDQ) model achieves the best performance in the vast majority of data precision scenarios. For example, the Q-A^{2}NN-ResNet-18-6bit model (SSDQ) outperforms the Q-CNN-ResNet-18-6bit model and the Q-A^{2}NN-ResNet-18-6bit model (POT) by 0.97%/3.15%, 1.08%/3.36%, and 1.21%/3.36% in Pre, Rec, and F1, respectively. Notably, the A^{2}NN-ResNet-18 model performs lower than the CNN-ResNet-18 model by 0.32% in Pre, Rec, and F1 metrics. This demonstrates that the Q-A^{2}NN obtained through the proposed SSDQ method can achieve more comprehensive performance. Moreover, in most cases, the Q-A^{2}NN (SSDQ) models show the least performance degradation compared to the floating-point precision model, and in some instances, even a slight improvement, at the same data precision. Overall, the above quantitative analysis proves the effectiveness of the proposed SSDQ method.

Table 4. Comparison results on the WHU data sets [Pre/Rec/F1 (%)].

Backbone	Precision	Q-CNN [54,55]	Q-A^{2}NN (POT) (Ours)	Q-A^{2}NN (SSDQ) (Ours)
ResNet-18	Floating-point	92.13/91.23/91.27	91.81/90.91/90.95	91.81/90.91/90.95
	8-bit	91.57/90.75/90.76	92.10/90.43/90.83	93.08/92.31/92.47
	6-bit	91.58/90.75/90.76	89.40/88.47/88.61	92.55/91.83/91.97
	4-bit	90.54/89.74/89.79	4.40/10.37/5.0	91.38/90.96/91.0
VGGNet-11	Floating-point	92.50/91.75/91.84	92.93/92.33/92.15	92.93/92.33/92.15
	8-bit	91.31/90.84/90.64	91.36/90.89/90.35	92.01/91.89/91.58
	6-bit	90.52/90.35/90.27	83.79/80.30/80.13	91.90/91.41/91.22
	4-bit	91.22/90.48/90.18	4.47/9.39/3.54	91.88/91.48/91.23

Figure 7 shows a comprehensive performance comparison of different models on the WHU data set. Figure 7a depicts the variation in the OA for quantization models with different quantization bit widths. The Q-A^{2}NN model sets (POT), obtained solely using the proposed POT-based shared scaling factor quantization scheme, still demonstrated excellent classification performance at 7-bit quantization. However, as the quantization bit width decreased, the performance of the Q-A^{2}NN model sets (POT) deteriorated sharply. In contrast, the Q-A^{2}NN model sets (SSDQ) and the Q-CNN model sets maintained excellent classification performance, even at low quantization bit widths. Moreover, the classification accuracies of the Q-A^{2}NN model sets (SSDQ) even surpassed those of the A^{2}NN and CNN models for some quantization bit widths, achieving the best classification performance among these models. Figure 7b displays the OA against the computational overhead and memory overhead for different models. The computational overhead of each model is represented by the size of the circle. While the full-precision CNN model, positioned in the top right corner of the figure, demonstrated outstanding classification performance, it entails a substantial number of parameters and has high computational demand. The AdderNet and A^{2}NN models effectively reduced the computational overhead, and the A^{2}NN model achieved comparable classification performance to the CNN model. Nevertheless, the memory overhead of the A^{2}NN model remains significant. Although the BNN model exhibited the smallest computational and memory overheads, its classification performance was insufficient for practical applications. The quantized model sets reduce the memory and computational overheads by converting the floating-point operations into low-bit-width integer operations. Among these quantized model sets, the Q-A^{2}NN model sets (SSDQ), positioned in the top left corner, achieved similar classification accuracy as the full-precision CNN model with reduced computational and memory overheads. As a result, the resource-efficient and high-performance Q-A^{2}NN model sets (SSDQ) are more suitable for edge hardware deployment than the other models. Notably, depending on the classification performance and resource overhead requirements in diverse application scenarios, a suitable model among the Q-A^{2}NN model sets (SSDQ) can be selected for onboard RSSC.

Figure 8 presents the confusion matrices of six different models. Among them, the Q-A^{2}NN-VGGNet-11-6bit model (SSDQ) presented the best performance. Specifically, the accuracies for all classes of the Q-A^{2}NN-VGGNet-11-6bit model exceeded 80%, and the accuracies for more than half of the classes were greater than 90%. The BNN-VGGNet-11 and AdderNet-VGGNet-11 models achieved accuracies exceeding 80% for only 6 and 10 classes, respectively. Additionally, the CNN-VGGNet-11 and Q-CNN-VGGNet-11-6bit models demonstrated accuracies exceeding 90% for only three classes.

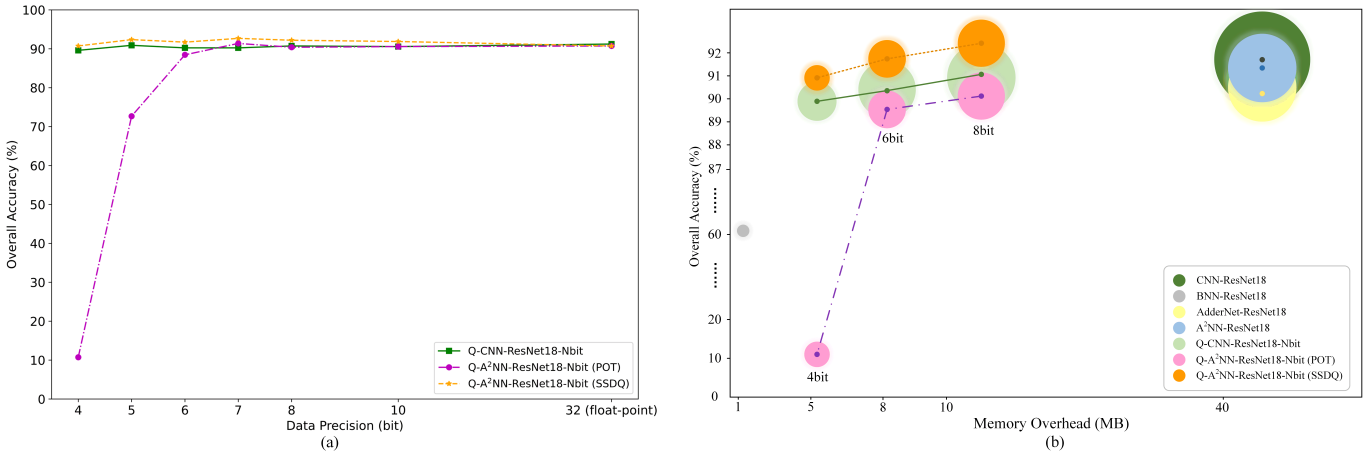


Figure 7. Comprehensive performance comparison of different models on the WHU data set: (**a**) OA of the Q-CNN-ResNet-18-Nbit models and Q-A^{2}NN-ResNet-18-Nbit models (SSDQ); and (**b**) comparisons of the OA and resource overhead of various models.

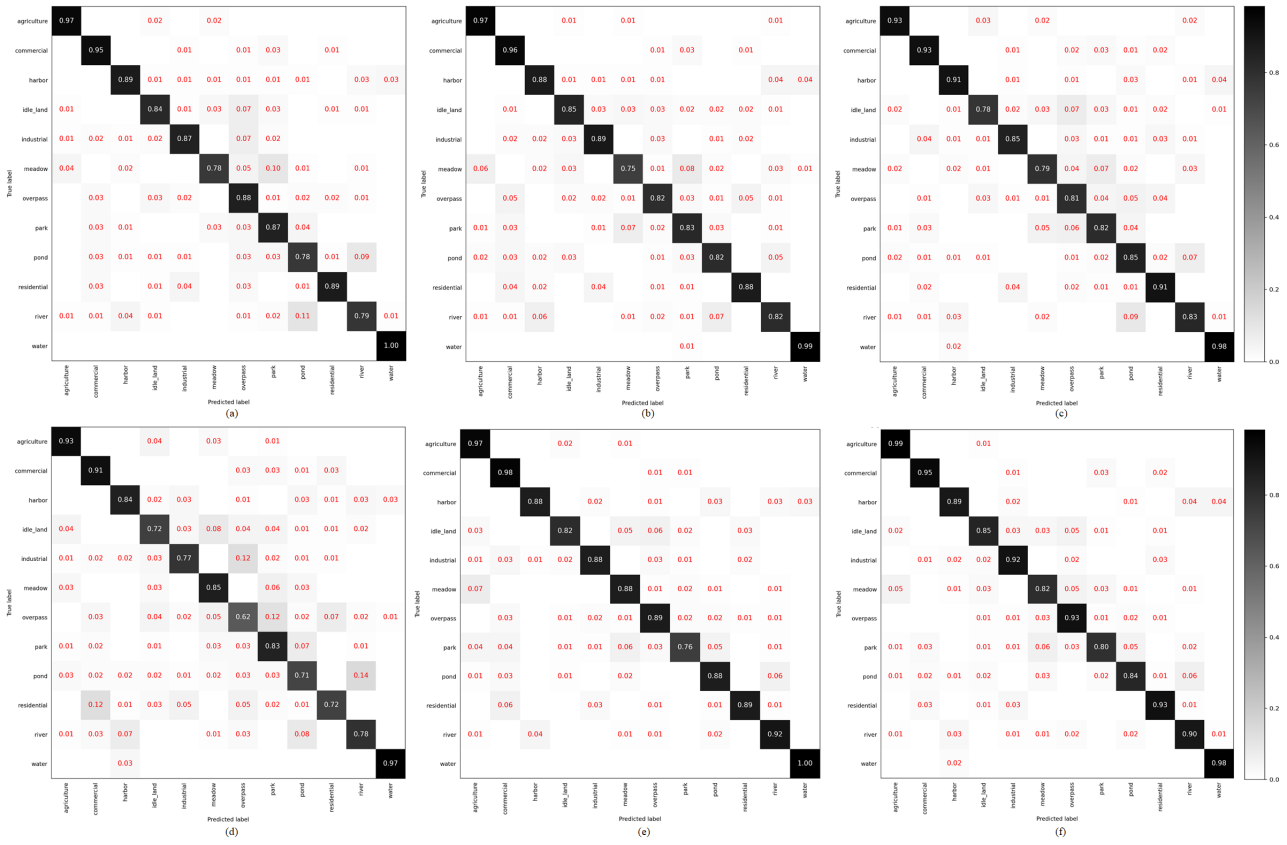


Figure 8. Confusion matrices of different models on the SIRI-WHU data set: (**a**) The CNN-VGGNet-11 model; (**b**) the Q-CNN-VGGNet-11-6bit model; (**c**) the AdderNet-VGGNet-11 model; (**d**) the BNN-VGGNet-11 model; (**e**) the A^{2}NN-VGGNet-11 model; and (**f**) the Q-A^{2}NN-VGGNet-11-6bit model (SSDQ).

5.6. Ablation Studies

We designed four variants to obtain Q-A^{2}NN model sets for ablation studies. The OAs of the Q-A^{2}NN model sets quantized with different component combinations of the proposed SSDQ method are displayed in Table 5.

Table 5. Ablation study on the proposed SSDQ method [OA ± STD (%)].

Data Set	Backbone	Basic Network	Precision						
			Floating-Point	10-bit	8-bit	7-bit	6-bit	5-bit	4-bit
UCM	ResNet-18	A^2NN	95.71	-	-	-	-	-	-
		Q-A^2NN (POT)	-	95.56 ± 0.13	94.29 ± 0.24	92.54 ± 0.14	88.02 ± 0.27	11.98 ± 1.79	11.98 ± 1.79
		Q-A^2NN (POT + FD)	-	95.95 ± 0	95.24 ± 0	94.13 ± 0.36	90.48 ± 1.03	40.32 ± 2.08	12.94 ± 1.45
		Q-A^2NN (POT + WD)	-	95.56 ± 0.13	94.29 ± 0.24	94.29 ± 0.24	93.97 ± 0.60	94.05 ± 0.24	93.57 ± 0.24
		Q-A^2NN (SSDQ)	-	95.95 ± 0.24	95.40 ± 0.14	95 ± 0	95.40 ± 0.14	95.24 ± 0	95.16 ± 0.14
	VGGNet-11	A^2NN	96.9	-	-	-	-	-	-
		Q-A^2NN (POT)	-	96.75 ± 0.13	95.48 ± 0	91.11 ± 0.14	67.54 ± 0.60	10.08 ± 0.72	10.16 ± 0.90
		Q-A^2NN (POT + FD)	-	96.67 ± 0	96.98 ± 0.36	94.76 ± 0.24	87.03 ± 0.84	11.67 ± 2.16	10.24 ± 0.24
		Q-A^2NN (POT + WD)	-	96.75 ± 0.13	95.48 ± 0	95.79 ± 0.14	96.35 ± 0.28	95.79 ± 0.14	94.60 ± 0.60
		Q-A^2NN (SSDQ)	-	96.67 ± 0	97.14 ± 0.24	96.51 ± 0.14	96.51 ± 0.14	96.03 ± 0.14	94.84 ± 0.28
WHU	ResNet-18	A^2NN	90.73	-	-	-	-	-	-
		Q-A^2NN (POT)	-	90.57 ± 0.28	90.40 ± 0.28	91.38 ± 0.28	88.45 ± 0.28	72.68 ± 0	10.73 ± 0
		Q-A^2NN (POT + FD)	-	91.87 ± 0.28	92.20 ± 0	92.68 ± 0.49	90.57 ± 0.74	79.51 ± 0.98	10.08 ± 0.28
		Q-A^2NN (POT + WD)	-	90.73 ± 0	90.24 ± 0	91.38 ± 0.28	89.27 ± 0	91.55 ± 0.28	90.73 ± 0
		Q-A^2NN (SSDQ)	-	91.87 ± 0.28	92.20 ± 0	92.68 ± 0.49	91.71 ± 0	92.36 ± 0.28	90.73 ± 0
	VGGNet-11	A^2NN	92.2	-	-	-	-	-	-
		Q-A^2NN (POT)	-	91.54 ± 0.29	90.89 ± 0.28	87.48 ± 0.28	80.49 ± 0.49	9.92 ± 0.28	9.76 ± 0
		Q-A^2NN (POT + FD)	-	91.71 ± 0	91.71 ± 0	89.11 ± 0.28	86.01 ± 0.28	10.08 ± 0.28	9.76 ± 0
		Q-A^2NN (POT + WD)	-	91.87 ± 0.28	90.89 ± 0.28	87.48 ± 0.28	90.40 ± 0.28	92.04 ± 0.28	91.06 ± 0.28
		Q-A^2NN (SSDQ)	-	91.71 ± 0	91.71 ± 0	89.11 ± 0.28	91.22 ± 0	91.71 ± 0	91.22 ± 0.49
RSSCN7	ResNet-18	A^2NN	82.9	-	-	-	-	-	-
		Q-A^2NN (POT)	-	83.68 ± 0.05	83.20 ± 0.09	82.40 ± 0.07	70.61 ± 0.14	40.61 ± 0.13	20.64 ± 0.20
		Q-A^2NN (POT + FD)	-	83.30 ± 0.14	83.21 ± 0.05	83.24 ± 0.05	77.90 ± 0.12	50.25 ± 0.14	24.84 ± 0.22
		Q-A^2NN (POT + WD)	-	83.68 ± 0.05	83.20 ± 0.09	82.40 ± 0.07	83.32 ± 0.32	81.90 ± 0.07	79.70 ± 0.27
		Q-A^2NN (SSDQ)	-	83.36 ± 0.11	83.36 ± 0.05	83.29 ± 0.17	83.18 ± 0.11	82.69 ± 0.02	81.40 ± 0.18
	VGGNet-11	A^2NN	83.08	-	-	-	-	-	-
		Q-A^2NN (POT)	-	84.49 ± 0.29	83.62 ± 0.28	81.13 ± 0.28	54.40 ± 0.49	18.32 ± 0.28	17.49 ± 0
		Q-A^2NN (POT + FD)	-	84.27 ± 0.07	83.91 ± 0.14	82.75 ± 0.13	72.57 ± 0.33	19.70 ± 0.05	18.07 ± 0.05
		Q-A^2NN (POT + WD)	-	84.50 ± 0.07	83.63 ± 0.07	81.07 ± 0.05	84.55 ± 0.15	82.72 ± 0.09	80.77 ± 0.61
		Q-A^2NN (SSDQ)	-	84.30 ± 0.09	83.91 ± 0.14	82.75 ± 0.13	83.96 ± 0.09	83.16 ± 0.05	81.09 ± 0.30

First, only using the proposed POT-based shared scaling factor quantization scheme, the Q-A^2NN model sets (POT) exhibited remarkable classification performance, even when subjected to 7-bit quantization. Nonetheless, as the quantization bit width decreased, the performance of the Q-A^2NN model sets (POT) sharply deteriorated. This performance decline can be attributed to the deviations in the quantized weights and output features of each layer due to the skewed weight distribution and accumulation of quantization errors.

Next, the WD and FD strategies in the MDD quantization strategy were independently introduced as variants to verify their respective effects. As illustrated in Table 5, each of the two proposed strategies positively impacted the classification performance of the Q-A^2NN model sets. The FD strategy enhances the performance of the Q-A^2NN model sets by reducing the deviation among the output features of each layer. For example, compared to the Q-A^2NN model sets (POT), for the ResNet-18 backbone, the Q-A^2NN model sets (POT + FD) improved the OA by approximately 0.39–28.34% on the UCM data set; for the VGGNet-11 backbone, the Q-A^2NN model sets (POT + FD) improved the OA by approximately 0.08–19.49% on the UCM data set. However, at 4-bit and 5-bit quantization, the classification performance of the Q-A^2NN model sets (POT + FD) was insufficient for practical applications. This issue is caused by the significant deviations in the low-bit-width quantized weights. The WD strategy can mitigate the severe performance degradation of the Q-A^2NN model sets by reducing the deviations in the quantized weights. The performance of the Q-A^2NN model sets (POT + WD) with low-bit-width quantization was significantly improved, compared with the baseline. For example, compared to the Q-A^2NN-4bit models (POT), the Q-A^2NN-ResNet-18-4bit model (POT + WD) improved the OA by approximately 80%, 59.06%, and 81.59% on the WHU, RSSCN7, and UCM data sets, respectively, and the Q-A^2NN-VGGNet-11-4bit model (POT + WD) improved the OA by approximately 84.44%, 81.30%, and 63.28% on the UCM, WHU, and RSSCN7 data sets, respectively. However, the classification accuracy of the Q-A^2NN model sets (POT + WD) still had a certain discrepancy, compared with that of the A^2NN models, on some data sets.

Finally, the Q-A^{2}NN model sets (SSDQ) achieved the best classification results across the above variants at most quantization bit widths. The ablation study results support the efficacy of the SSDQ method tailored for the quantization of A^{2}NNs. The proposed SSDQ method presents an efficient and effective solution to attain Q-A^{2}NN models with minimal performance degradation for onboard RSSC.

5.7. Visualization Analysis

To explain and verify that the proposed WD strategy can prevent the significant information loss caused by deviations in the quantized weights, parameter count histograms were employed to visualize the weight distributions of the adder filters or quantized adder filters in different models.

As shown in Figure 9a,b, for the A^{2}NN models, the weight distribution of the adder filter was wide and skewed. The weight distribution ranged from -50 to 90 in the 16th adder layer in the A^{2}NN-ResNet-18 model, while most weights were distributed between -2 and 2, accounting for over 80%. In contrast, the weights with values below -25 or above 50 were rare, accounting for less than 0.1%. The weight distribution in the third adder layer in the A^{2}NN-VGGNet model presented the same phenomenon. The quantized adder filters cause many weights that are densely distributed around zero to be compressed to the integer zero after quantization, as illustrated in Figure 9c,d. For 4-bit quantization, the quantized weight can be represented by 15 integer values from -7 to 7. However, whether in the 16th adder layer of the Q-A^{2}NN-ResNet-18-4bit model (POT) or the third adder layer of the Q-A^{2}NN-VGGNet-11-4bit model (POT), more than 60% of the quantized weight values were equal to zero. In this case, most of the feature extraction ability of the full-precision A^{2}NN model after quantization was lost, resulting in severely decreased accuracy. Notably, the above situation can be avoided by applying the proposed WD strategy. Figure 9e,f show that the quantized weight distributions were uniform after introducing the WD strategy. The number of quantization weights with values equal to zero in the 16th adder layer in the Q-A^{2}NN-ResNet-18-4bit model (POT + WD) was reduced by 70%; similarly, the number of quantization weights with values equal to zero in the third adder layer in the Q-A^{2}NN-VGGNet-11-4bit model (POT + WD) was reduced by 75%. In this way, the feature extraction ability possessed by the well-trained A^{2}NN models can be preserved as much as possible after the quantization process.

Moreover, the gradient-weighted class activation mapping (Grad-CAM) method [56] was employed to visually assess the impact of the proposed SSDQ method. The resulting class activation map (CAM) is presented as a thermal map, illustrating how the models focus on specific regions of the input image. In the visual representation, regions with stronger responses are highlighted in red, whereas regions with lower responses are depicted in blue. Figure 10 shows the CAM results for eight classes of images randomly selected from the five public RSSC data sets. From top to bottom is the input image, the CAM corresponding to the A^{2}NN-VGGNet-11 model, the CAM corresponding to the Q-A^{2}NN-VGGNet-11-4bit model (POT), the CAM corresponding to the Q-A^{2}NN-VGGNet-11-4bit model (POT + WD), and the CAM corresponding to the Q-A^{2}NN-VGGNet-11-4bit model (SSDQ). The CAM generated by the Q-A^{2}NN-VGGNet-11-4bit model (POT) did not effectively capture critical information in the remote sensing images, as the feature extraction ability possessed by the well-trained A^{2}NN model was severely degraded due to the deviations in the quantized weights and features caused by low-bit-width quantization. With the introduction of the WD strategy, the CAM generated by the Q-A^{2}NN-VGGNet-11-4bit model (POT + WD) could successfully concentrate on the crucial regions in most scene classes. However, it can still be observed, from the CAM results of certain scene classes, that critical parts (e.g., storage tanks and rivers) were not fully covered or even could not be extracted accurately, indicating a certain discrepancy compared with the A^{2}NN model. In contrast, the Q-A^{2}NN-VGGNet-11-4bit model (SSDQ) obtained through incorporating the proposed SSDQ method comprehensively covered the critical parts in the remote sensing scenes while minimizing interference from intricate backgrounds. Taking the center and

tennis court as examples, the Q-A^{2}NN-VGGNet-11-4bit model (SSDQ) located the critical regions more comprehensively than the A^{2}NN model. The above analysis indicates that the classification performance of Q-A^{2}NN models can be significantly enhanced through the use of the SSDQ method.

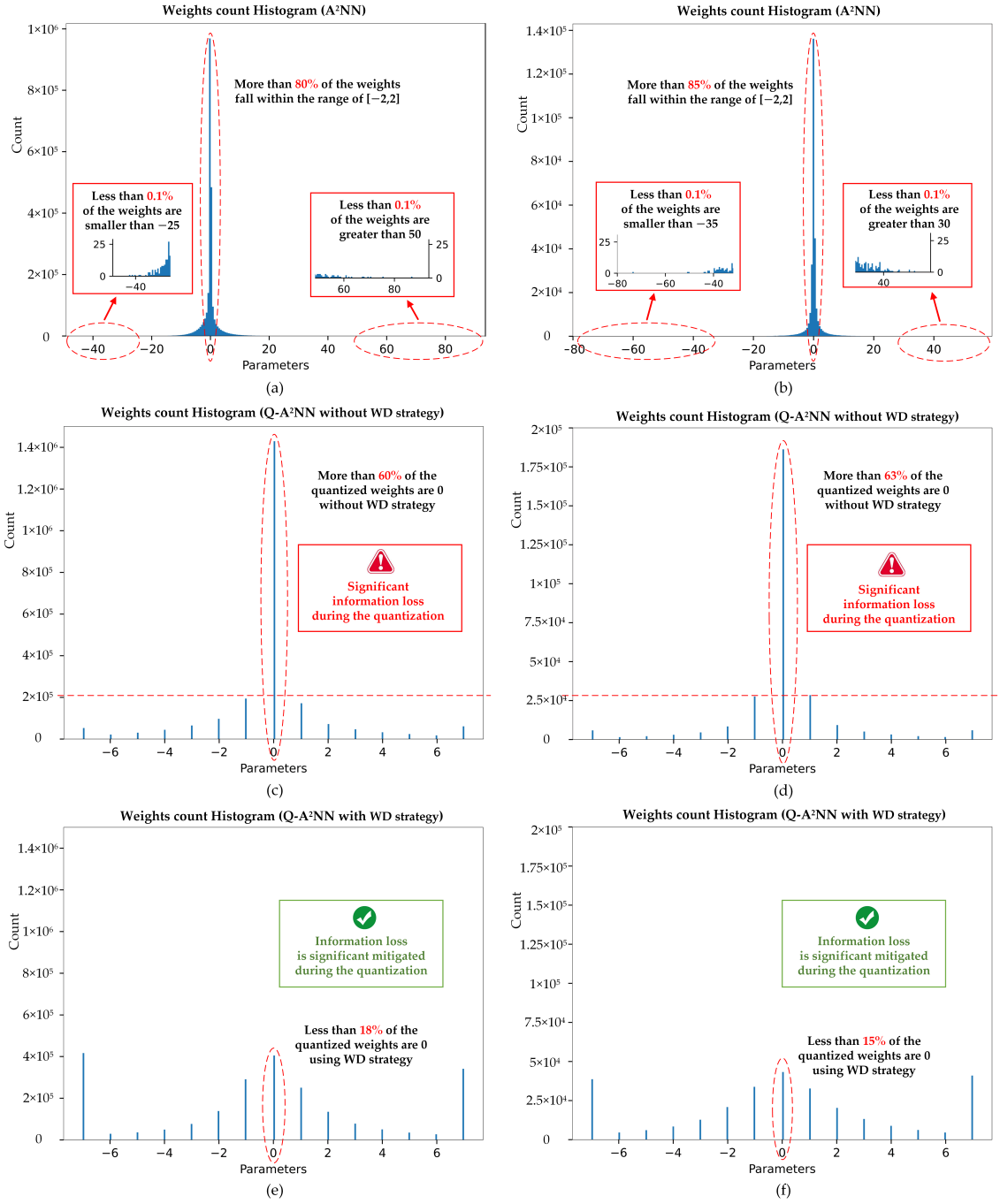


Figure 9. Parameter count histograms of the adder filter or quantized adder filters in different models on the AID data set: (**a**) The 16th layer in the A^{2}NN-ResNet-18 model; (**b**) the third layer in the A^{2}NN-VGGNet-11 model; (**c**) the 16th layer in the Q-A^{2}NN-ResNet-18-4bit model (POT); (**d**) the third layer in the Q-A^{2}NN-VGGNet-11-4bit model (POT); (**e**) the 16th layer in the Q-A^{2}NN-ResNet-18-4bit model (POT + WD); and (**f**) the third layer in the Q-A^{2}NN-VGGNet-11-4bit model (POT + WD).

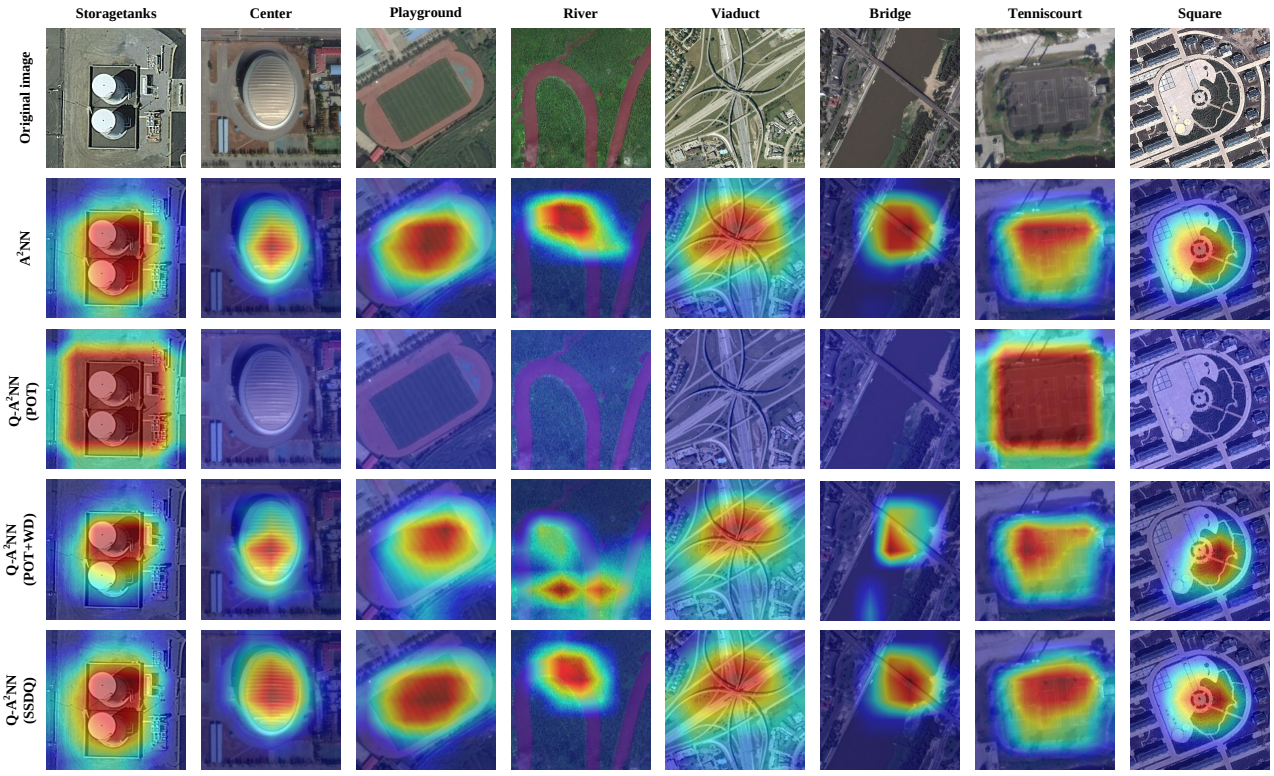


Figure 10. Visualization of the CAM results for eight scene classes of images randomly selected from the five public RSSC data sets.

6. Conclusions

This article proposed an SSDQ method tailored for quantizing A^2NNs, including a POT-based shared scaling factor quantization scheme and an MDD quantization strategy. The POT-based shared scaling factor quantization scheme is devised to quantize the adder filters, converting the adder filters in A^2NNs to quantized adder filters with hardware-friendly integer input activations, weights, and operations. Thus, Q-A^2NNs composed of quantized adder filters have lower computational and memory overheads than A^2NNs during hardware deployment. The MDD quantization strategy is formulated to avoid the performance degradation of Q-A^2NNs due to deviations in the weights and features during the quantization process. The MDD strategy synergistically integrates the WD strategy, which mitigates performance degradation stemming from deviations in the quantized weights, and the FD strategy, which enhances the classification performance of Q-A^2NNs by minimizing the deviations among the output features of each layer. Extensive experimentation and analysis on five commonly used RSSC data sets revealed that Q-A^2NN models with low computational overhead, low memory overhead, and minimal performance degradation can be obtained with the proposed SSDQ method. In particular, the obtained Q-A^2NN models are more suitable for onboard RSSC than CNNs and A^2NNs. In future work, we will attempt to enhance the performance of Q-A^2NN models by accounting for the complexity of spatial distributions in remote sensing images. Additionally, we plan to investigate the use of high-throughput and energy-efficient field-programmable gate array (FPGA)-based accelerators for Q-A^2NN models to fulfill the requirements of on-board real-time RSSC.

Author Contributions: N.Z. designed the model, then implemented the model and wrote the paper. H.C. and L.C. contributed to the supervision of the work, analysis of the method, and paper writing. J.W., G.W. and W.L. contributed to the analysis of the method, and paper writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation for Young Scientists of China (under Grant 62201059), in part by the Foundation (under Grant JCKY2021602B037), and in part by the BIT Research and Innovation Promoting Project (under Grant No. 2023YCXY006).

Data Availability Statement: The data used in this study are available upon request from the corresponding author due to privacy restrictions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, X.; Hong, D.; Chanussot, J. Convolutional neural networks for multimodal remote sensing data classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5517010. [CrossRef]
2. Du, X.; Zheng, X.; Lu, X.; Doudkin, A.A. Multisource remote sensing data classification with graph fusion network. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 10062–10072. [CrossRef]
3. Cao, X.; Yao, J.; Xu, Z.; Meng, D. Hyperspectral image classification with convolutional neural network and active learning. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 4604–4616. [CrossRef]
4. Wang, W.; Chen, Y.; Ghamisi, P. Transferring CNN With Adaptive Learning for Remote Sensing Scene Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5533918. [CrossRef]
5. Tong, W.; Chen, W.; Han, W.; Li, X.; Wang, L. Channel-attention-based DenseNet network for remote sensing image scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 4121–4132. [CrossRef]
6. Hong, D.; Gao, L.; Yokoya, N.; Yao, J.; Chanussot, J.; Du, Q.; Zhang, B. More diverse means better: Multimodal deep learning meets remote-sensing imagery classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 4340–4354. [CrossRef]
7. Grøtte, M.E.; Birkeland, R.; Honoré-Livermore, E.; Bakken, S.; Garrett, J.L.; Prentice, E.F.; Sigernes, F.; Orlandić, M.; Gravdahl, J.T.; Johansen, T.A. Ocean color hyperspectral remote sensing with high resolution and low latency—The hypso-1 cubesat mission. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1000619. [CrossRef]
8. Caba, J.; Díaz, M.; Barba, J.; Guerra, R.; de la Torre, J.A.; López, S. Fpga-based on-board hyperspectral imaging compression: Benchmarking performance and energy efficiency against gpu implementations. *Remote Sens.* **2020**, *12*, 3741. [CrossRef]
9. Wiehle, S.; Mandapati, S.; Günzel, D.; Breit, H.; Balss, U. Synthetic aperture radar image formation and processing on an MPSoC. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5226814. [CrossRef]
10. Zhang, B.; Wu, Y.; Zhao, B.; Chanussot, J.; Hong, D.; Yao, J.; Gao, L. Progress and challenges in intelligent remote sensing satellite systems. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 1814–1822. [CrossRef]
11. Fu, C.; Cao, Z.; Li, Y.; Ye, J.; Feng, C. Onboard real-time aerial tracking with efficient Siamese anchor proposal network. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5606913. [CrossRef]
12. Jaderberg, M.; Vedaldi, A.; Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *arXiv* **2014**, arXiv:1405.3866.
13. Zhang, X.; Zou, J.; Ming, X.; He, K.; Sun, J. Efficient and accurate approximations of nonlinear convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1984–1992.
14. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 7–12 December 2015; Volume 28.
15. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
16. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
17. Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P. Deep learning with limited numerical precision. In Proceedings of the International Conference on Machine Learning, Lille, France, 6 July–11 July 2015; pp. 1737–1746.
18. Lin, S.; Ji, R.; Chen, C.; Tao, D.; Luo, J. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2889–2905. [CrossRef] [PubMed]
19. Huang, Z.; Wang, N. Data-driven sparse structure selection for deep neural networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 304–320.
20. Zhang, Y.; Zhen, Y.; He, Z.; Yen, G.G. Improvement of efficiency in evolutionary pruning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
21. Liu, Y.; Cao, J.; Li, B.; Yuan, C.; Hu, W.; Li, Y.; Duan, Y. Knowledge distillation via instance relationship graph. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 7096–7104.
22. Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; Reid, I. Towards effective low-bitwidth convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7920–7928.
23. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.

24. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 9–13 February 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 10–14.
25. Wang, Y.; Huang, M.; Han, K.; Chen, H.; Zhang, W.; Xu, C.; Tao, D. AdderNet and its minimalist hardware design for energy-efficient artificial intelligence. *arXiv* **2021**, arXiv:2101.10015.
26. Valueva, M.V.; Nagornov, N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math. Comput. Simul.* **2020**, *177*, 232–243. [CrossRef]
27. Chen, H.; Wang, Y.; Xu, C.; Shi, B.; Xu, C.; Tian, Q.; Xu, C. AdderNet: Do we really need multiplications in deep learning? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1468–1477.
28. Zhang, N.; Wang, G.; Wang, J.; Chen, H.; Liu, W.; Chen, L. All Adder Neural Networks for On-board Remote Sensing Scene Classification. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5607916. [CrossRef]
29. Zhang, Y.; Sun, B.; Jiang, W.; Ha, Y.; Hu, M.; Zhao, W. WSQ-AdderNet: Efficient Weight Standardization based Quantized AdderNet FPGA Accelerator Design with High-Density INT8 DSP-LUT Co-Packing Optimization. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design, San Diego, CA, USA, 29 October–3 November 2022; pp. 1–9.
30. Li, H.; Gu, H.; Han, Y.; Yang, J. Object-oriented classification of high-resolution remote sensing imagery based on an improved colour structure code and a support vector machine. *Int. J. Remote Sens.* **2010**, *31*, 1453–1470. [CrossRef]
31. Cheng, G.; Han, J.; Lu, X. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE* **2017**, *105*, 1865–1883. [CrossRef]
32. Zhu, Q.; Zhong, Y.; Zhao, B.; Xia, G.S.; Zhang, L. Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 747–751. [CrossRef]
33. Zhao, B.; Zhong, Y.; Xia, G.S.; Zhang, L. Dirichlet-derived multiple topic scene classification model for high spatial resolution remote sensing imagery. *IEEE Trans. Geosci. Remote Sens.* **2015**, *54*, 2108–2123. [CrossRef]
34. Wang, S.; Guan, Y.; Shao, L. Multi-granularity canonical appearance pooling for remote sensing scene classification. *IEEE Trans. Image Process.* **2020**, *29*, 5396–5407. [CrossRef] [PubMed]
35. Wang, Q.; Huang, W.; Xiong, Z.; Li, X. Looking closer at the scene: Multiscale representation learning for remote sensing image scene classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *33*, 1414–1428. [CrossRef]
36. Sun, H.; Li, S.; Zheng, X.; Lu, X. Remote sensing scene classification by gated bidirectional network. *IEEE Trans. Geosci. Remote Sens.* **2019**, *58*, 82–96. [CrossRef]
37. Wang, J.; Chen, H.; Ma, L.; Chen, L.; Gong, X.; Liu, W. Sphere Loss: Learning Discriminative Features for Scene Classification in a Hyperspherical Feature Space. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5601819. [CrossRef]
38. Bazi, Y.; Bashmal, L.; Rahhal, M.M.A.; Dayil, R.A.; Ajlan, N.A. Vision transformers for remote sensing image classification. *Remote Sens.* **2021**, *13*, 516. [CrossRef]
39. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
40. Sha, Z.; Li, J. MITformer: A multiinstance vision transformer for remote sensing scene classification. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 6510305. [CrossRef]
41. Nagel, M.; Fournarakis, M.; Amjad, R.A.; Bondarenko, Y.; Van Baalen, M.; Blankevoort, T. A white paper on neural network quantization. *arXiv* **2021**, arXiv:2106.08295.
42. Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [CrossRef]
43. Yuan, Z.; Xue, C.; Chen, Y.; Wu, Q.; Sun, G. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In Proceedings of the Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Part XII; Springer: Cham, Switzerland, 2022; pp. 191–207.
44. Li, Z.; Li, X.; Yang, L.; Zhao, B.; Song, R.; Luo, L.; Li, J.; Yang, J. Curriculum temperature for knowledge distillation. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 1504–1512.
45. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
46. Sheng, G.; Yang, W.; Xu, T.; Sun, H. High-resolution satellite scene classification using a sparse coding based multiple feature combination. *Int. J. Remote Sens.* **2012**, *33*, 2395–2412. [CrossRef]
47. Yang, Y.; Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; pp. 270–279.
48. Zou, Q.; Ni, L.; Zhang, T.; Wang, Q. Deep learning based feature selection for remote sensing scene classification. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2321–2325. [CrossRef]
49. Xia, G.S.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. AID: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3965–3981. [CrossRef]

50. Hu, Y.; Huang, X.; Luo, X.; Han, J.; Cao, X.; Zhang, J. Variational Self-Distillation for Remote Sensing Scene Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5627313. [CrossRef]
51. Xu, K.; Deng, P.; Huang, H. Vision Transformer: An Excellent Teacher for Guiding Small Networks in Remote Sensing Image Scene Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5618715. [CrossRef]
52. Courbariaux, M.; Bengio, Y.; David, J.P. Binaryconnect: Training deep neural networks with binary weights during propagations. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 7–12 December 2015; Volume 28.
53. Loshchilov, I.; Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv* **2016**, arXiv:1608.03983.
54. Wei, X.; Chen, H.; Liu, W.; Xie, Y. Mixed-precision quantization for CNN-based remote sensing scene classification. *IEEE Geosci. Remote Sens. Lett.* **2020**, *18*, 1721–1725. [CrossRef]
55. Wei, X.; Liu, W.; Chen, L.; Ma, L.; Chen, H.; Zhuang, Y. FPGA-based hybrid-type implementation of quantized neural networks for remote sensing applications. *Sensors* **2019**, *19*, 924. [CrossRef]
56. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.