Improving Post-Training Quantization via Probabilistic Programming

Kui Liu, Bart Goossens, *Member*, IEEE, Tom De Schepper, Wilfried Philips, *Senior Member*, IEEE

***Abstract*—Post-training quantization (PTQ) is an effective solution for deploying deep neural networks on edge devices with limited resources. PTQ is especially attractive because it does not require access to the entire original training dataset on the promise of being able to use a much smaller calibration dataset. However, many existing PTQ methods still require a sufficiently large calibration dataset (e.g., more than 1000 images) to achieve satisfactory model accuracy. In this paper, we present a novel post-training quantization method that estimates quantization parameters using a Bayesian Maximum A Posterior (MAP) estimator. By modeling the uncertainty of quantization operations, we formulate the neural network quantization as a Bayesian inference problem. In our method, we first employ probabilistic programming techniques to optimize quantization parameters by maximizing the posterior of quantization step sizes. In addition, we introduce a Minimum Description Length (MDL) prior that favors low quantization bit widths and a validation procedure, which enhances PTQ performance when learning from small calibration datasets. Comprehensive evaluations demonstrate that the proposed method can improve the PTQ performance using a minimal calibration dataset of just 64 images, and achieve nearly state-of-the-art PTQ performance. Furthermore, the proposed method shows strong generalization ability when calibrated on different data sources and tested across diverse data.**

***Index Terms*—Post-training quantization, Bayesian optimization, probabilistic programming**

I. Introduction

The deep neural networks have achieved tremendous success in a wide range of domains. However, their significant storage and computational requirements have hindered deployment on most mobile platforms due to limited computational and storage resources. To address this challenge, neural network compression have drawn increased attentions, including pruning [1], low-rank factorization [2], knowledge distillation [3], and quantization [4]. Quantization is one of the most widely used compression techniques, which can be applied to various hardware like general CPU and GPU. By replacing floating-point values with low precision fixed-point representation, quantization not only significantly reduces the size of data but also enables more efficient integer additions and multiplications on hardware platforms.

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

Kui Liu, Bart Goossens, Wilfried Philips are with the Image Processing and Interpretation Research Group, Interuniversity Microelectronics Centre (IMEC), Ghent University, 9000, Ghent, Belgium (e-mail: kui.liu@ugent.be; bart.goossens@ugent.be; wilfried.philips@ugent.be).

Tom De Schepper is with AI & Data Department, Interuniversity Microelectronics Centre (IMEC), 3001, Leuven, Belgium (e-mail: tom.deschepper@imec.be).

However, the direct conversion from floating-point to fixed-point involves rounding operations, which generally degrades the accuracy of pre-trained models since it crudely rounds values (e.g., neural network weights and activations) to the nearest integers. To mitigate this degradation, many existing quantization approaches employ Quantization-Aware Training (QAT) methods, which utilize a retraining process to finetune the pre-trained weights or quantization parameters. The QAT methods have achieved impressive performance on low precision quantization [5], [6]. However, these methods typically require access to the full training dataset, substantial computational resources, and extended training time for the retraining process [7], [8]. Additionally, in some real-world scenarios, full access to training datasets is unavailable due to concerns for business security and privacy protection, which restricts QAT methods to broader applications.

Post-Training Quantization (PTQ) methods have been proposed to avoid the use of retraining. In the absence of a full training dataset, PTQ methods aim at tuning quantized models using a relatively small amount of calibration data. However, without retraining process, PTQ methods often suffer from significant accuracy degradation especially when the quantization precision is low [9]. To address this issue, local quantization error minimization has been proven to be an effective strategy to improve PTQ performance by determining appropriate quantization parameters [10], [11]. In recent studies, researchers have analyzed the loss degradation using Taylor series expansion and derived a layer-wise reconstruction objective to recover performance loss from quantization [12], [13]. Although these PTQ methods already use a calibration dataset that is much smaller than the original training dataset, we aim to reduce the calibration dataset size further with the motivation that it significantly facilitates model deployment (e.g., some medical datasets include a few hundred samples [14]). Compared to retraining methods that use a dataset similar in size to the original dataset, optimizing on a small calibration dataset is more prone to overfitting, resulting in poor generalization performance on test data. To address this issue, we employ a Bayesian approach to estimate appropriate quantization parameters by framing neural network quantization as a probabilistic inference problem. As indicated in works [15], [16], there are many promising successes in using probabilistic approaches on small datasets by incorporating domain knowledge into the models.

Several works use Bayesian estimation for neural network quantization [17]–[19]. In [17], the authors use variational posterior uncertainty to assess significant bits and remove those fluctuating too much under approximate posterior sam-

pling. Later, this method is extended by [18] that introduces a multi-modal quantizing prior to benefit weight quantization. Recently, the work [19] introduces learnable gates in quantization operations, enabling bit width optimization through variational inference. However, to achieve good quantization performance, these approaches involve a retraining process to fine-tune quantization parameters and weights.

As a flexible alternative to deep learning, probabilistic programming techniques [20], [21] have emerged to leverage the advantages of probabilistic models by expressing probabilistic inference as a computer program. This program specifies the process of generating output data by sampling latent probability distributions and can utilize the observed output data to optimize these latent distributions. The designed probabilistic models employ gradient-based methods to conduct straightforward optimization due to automatic differentiation. By overtly representing uncertainty and yielding explainable models, well-designed probabilistic models have shown great promise to learn from small data [16]. With the assistance of probabilistic programming techniques, it is more convenient to develop deep learning-based probabilistic applications.

In this study, we propose a novel principled PTQ method, which is named Probabilistic Programming Quantization (PPQ). We first employ an additive uniform noise model to approximate quantization operations in neural networks for PTQ, which facilitates the use of Bayesian optimization to infer the quantization parameters for uniform quantization. By treating quantization step sizes as random variables, we employ a hierarchical Bayesian model to probabilistically model the variables and quantization operations involved in neural network quantization. Through Monte Carlo approximation, we obtain a probabilistic optimization objective aimed at maximizing the posterior of quantization step sizes from the sense of Maximum A Posterior (MAP) inference.

The proposed MAP estimation is adapted to a differentiable objective with respect to quantization step sizes, enabling direct gradient-based optimization through probabilistic programming. In addition, we introduce a Minimum Description Length (MDL) prior that favors low quantization bit widths for quantization step sizes, which regularizes the growth of the estimated step sizes during the optimization process. We utilize a limited unlabeled calibration dataset to optimize the proposed MAP estimator. To mitigate overfitting on a small dataset, we employ a validation procedure based on best model checkpointing to estimate quantization step sizes on this small dataset. This procedure is combined with early stopping to select the quantization parameters. Comprehensive experiments demonstrate the method's robust performance across varying calibration dataset sizes and different data sources, enhancing the adaptability of PTQ methods for real-world deployment.

Our main contributions are summarized as follows:

- A novel Probabilistic Programming Quantization (PPQ) method is proposed by modeling the uncertainty of quantization and estimating the optimal quantization parameters through hierarchical Bayesian modeling.
- We introduce a Minimum Description Length (MDL) prior and validation procedure, which enhances quantization performance on small calibration datasets. In addition, we integrate a percentile-based clipping method to mitigate the effects of outliers in the weights and activations.
- The Bayesian problem for neural network quantization is formulated and adapted to be solved in probabilistic programming languages (in our case, TensorFlow Probability and Pyro), enabling a gradient-based optimization approach to maximize the posterior of quantization step sizes. To the best of our knowledge, this is the first time that the probabilistic programming is used for neural network quantization.
- When evaluated on different datasets (e.g., CIFAR10 and ImageNet) using various models, the proposed method achieves nearly state-of-the-art quantization performance with arbitrarily selected 64 calibration images, and exhibits good generalization ability to optimize with different data sources and strong adaptability to diverse data.

The organization of this paper is as follows: Section II provides a review of recent research on neural network quantization. Section III presents the proposed quantization method and introduces a new quantization range searching strategy. Section IV describes the implementation of probabilistic programming and optimization objectives. The experimental results and discussions are presented in Section V. Finally, Section VI concludes this paper.

II. Related Work

Neural network quantization has been gaining significant attention due to its ability to significantly reduce the memory usage and computational complexity of deep neural networks. Current quantization research can be broadly categorized into two classes: Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ).

A. Quantization-Aware Training

The QAT methods focus on fine-tuning neural network weights or quantization parameters to minimize performance loss through retraining on the entire training dataset. LQ-Nets [22] learns adaptive optimizers for weights and activations during the training process. PACT [7] modifies the activation function to enable learning the clipping ranges through back-propagation. LSQ [8] introduces a new gradient estimation method to learn quantization step sizes for non-negative activation functions. In [23], the authors adapt different bit width assignments during training for flexible deployments on diversified devices. By minimizing the quantization error on the entire training dataset, QAT methods have demonstrated significant performance improvements. These methods have even achieved decent accuracy with extremely low-precision quantization, e.g., 1-bit or 2-bit quantization [24], [25]. However, the QAT methods require access to a full training dataset and a time-consuming computation process, which hinders their broader application in some real-world scenarios.

B. Post-Training Quantization

The PTQ methods perform quantization optimization by adjusting quantized models without retraining. However, because PTQ methods do not use the original training dataset

for retraining, these methods often suffer from greater accuracy degradation compared to QAT methods. To mitigate accuracy degradation, some boosting strategies are commonly employed, such as bias correction [10], [26], weight equalization [26], and quantization range searching [13], [27].

Additionally, various optimization-based methods have been proposed to improve the post-training performance. Adaround [12] reveals that adaptive rounding can surpass the predominant rounding-to-nearest method by utilizing quadratic binary optimization to determine rounding up or rounding down. SPARQ [28] leverages dynamic activation sparsity in different representation granularities to pick the most significant n bits. BRECQ [13] demonstrates that block reconstruction can achieve better performance compared to a per-layer manner. Bit-Split [29] simultaneously optimizes the quantization scales as well as the low-bit weights to achieve minimal accuracy degradation. PFQ [30] mitigates the quantization error attributed to precision violation through a prepositive feature quantization framework. RDO-PTQ [31] improves quantization performance for image compression models by jointly considering rate-distortion loss.

However, a small calibration dataset brings challenges in overfitting during the optimization process. To address this issue, LAPQ [32] utilizes cross-layer dependency to jointly optimize the quantization parameters across all network layers. AdaQuant [33] suggests a modified objective for per-layer joint optimization of the weights and quantization parameters with the training loss serving as a penalty metric. QDROP [34] drops partial quantization during PTQ reconstruction to pursue a better performance on test data. PD-Quant [35] adjusts the distribution of activations to mitigate the overfitting problem. Nevertheless, these PTQ methods still require a calibration dataset including several hundred to a thousand images to maintain good quantization performance.

In contrast, this work formulates the quantization problem as a Bayesian estimation by modeling the uncertainty caused by the rounding operations. Incorporated with an MDL prior of quantization step sizes and a validation procedure, the proposed method achieves nearly state-of-the-art PTQ performance using a calibration dataset of fewer than 100 images.

C. Data-Free Quantization

Due to privacy or business concerns, it is sometimes not possible to release the training data for a neural network. Therefore, data-free (zero-shot) quantization has emerged as an important quantization research branch. However, these methods suffer from obvious degradation at low precision quantization [26], [36].

To bridge the gap between data-free quantization and data-driven quantization, many data generation-based quantization methods have been proposed recently. ZeroQ [37] demonstrates that synthetic data generated from batch normalization layer statistics matches the sensitivity of the original training dataset. Qimera [38] introduces latent embedding into the generator to produce synthetic boundary-supporting samples. KMDFQ [39] learns a knowledge matching generator to produce meaningful data. In [40], the authors encourage the generator to generate heterogeneous synthetic data by providing additional supervision information. These data generation-based quantization approaches eliminate the need for an original dataset. However, it is worth noting that for most DFQ methods, training on large generated datasets is still necessary [29]. While these methods alleviate the privacy concerns of the data, they do not necessarily reduce the computational demands of model quantization.

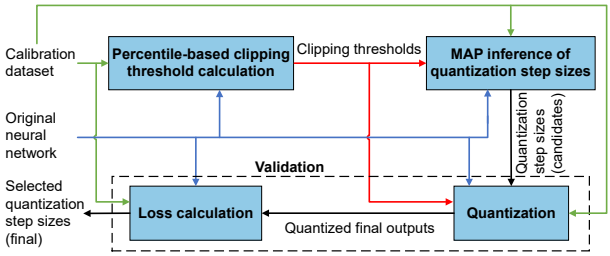

Fig. 1: The overview of the proposed quantization method.

III. Post-Training Quantization via Probabilistic Inference

In this section, we describe the principle of using Bayesian inference to improve the quantization performance. We model statistical fluctuations caused by the quantization of weights and activations with statistically independent additive uniform noise sources, allowing the use of the MAP inference to estimate optimal quantization parameters. Next, we introduce a percentile-based clipping method to mitigate the adverse effect caused by strong outliers in the weights and activations.

A. Overview

Fig. 1 shows the overview of the proposed method. We introduce a percentile-based clipping method to determine appropriate clipping thresholds before the probabilistic inference procedure. Afterwards, MAP inference is utilized to estimate the optimal quantization step sizes through the probabilistic programming technique [20], [21]. A validation procedure designed to mitigate overfitting is introduced during the optimization process to select the optimized quantization parameters, which will be further explained in Section IV.

B. Background Information

The goal of post-training quantization is to convert the floating-point weights and activations of neural networks into a low-precision representation, while minimizing the negative impact on the model's performance without needing to fine-tune weights using the full training dataset. To achieve this, an optimization objective is typically created to estimate optimal quantization parameters (e.g., quantization step sizes), which usually requires a calibration dataset [12], [13]. We compared the objective functions used in [12], [29], [41], and found that they are typically in the following form:

$$\arg\min_{\boldsymbol{\Phi}} \sum_{i=1}^{N} \left\| \mathbf{Y}_i^{\mathrm{nq}} - \mathrm{NN}(\mathbf{X}_i; \boldsymbol{\theta}, \boldsymbol{\phi}) \right\|^2, \tag{1}$$

where $\mathrm{NN}(\mathbf{X}_i; \boldsymbol{\theta}, \boldsymbol{\phi})$ represents the quantized neural network function with quantization parameters $\boldsymbol{\phi} = (\boldsymbol{\phi}_l)_{l=1}^{L} \in \mathbb{R}^H$ and neural network parameters (weights and biases) $\boldsymbol{\theta} = (\boldsymbol{\theta}_l)_{l=1}^{L} \in \mathbb{R}^D$. Here, L denotes the number of layers, while $H = \sum_{l=1}^{L} H_l$ and $D = \sum_{l=1}^{L} D_l$ represent the number of quantization parameters and the number of network parameters across all layers, respectively. Here, $\mathbf{X}_i \in \mathbb{R}^I$ represents the i-th input image and $\mathbf{Y}_i^{\mathrm{nq}} \in \mathbb{R}^O$ represents the output of the unquantized neural network on the input image $\mathbf{X}_i$, where I and O denote the dimension of a single input image and the dimension of the output vector for a single image, respectively. N denotes the number of the calibration samples.

In practice, a global optimization of Eq. 1 for PTQ is usually infeasible due to the high computational and memory complexity [13], [33], as well as the non-convexity of the objective functions [11], [12]. To tackle this problem, recent studies [12], [29] quantitatively analyze the loss degradation caused by quantization by viewing weight quantization as perturbation. A more practical layer-wise optimization objective is obtained using Taylor series expansions:

$$\arg\min_{\boldsymbol{\Phi}_l} \sum_{i=1}^{N} \left\| \mathcal{C}\left(\mathbf{W}^{(l)}, \mathbf{X}_i^{(l)}\right) - \mathcal{C}\left(Q_{\boldsymbol{\Phi}_l}(\mathbf{W}^{(l)}), \mathbf{X}_i^{(l)}\right) \right\|^2, \tag{2}$$

where $\mathcal{C}(\cdot)$ denotes a standard neural network convolutional operation. $\mathbf{X}_i^{(l)} \in \mathbb{R}^{A_l}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{J_l}$ represent the activations and weights of the l-th layer with A_l and J_l denoting their dimensions, respectively. $Q_{\boldsymbol{\Phi}_l}(\cdot)$ denotes the quantization function with quantization parameters $\boldsymbol{\phi}_l \in \mathbb{R}^{H_l}$. The optimization of activation quantization can be performed in an analogous way [29]. This objective function can be optimized through various approaches, e.g., integer programming [33], block-wise reconstruction [13], and optimization of bit weights [29]. The layer-wise optimization strategy effectively enables PTQ methods to learn from a small calibration dataset.

However, it is still difficult for the above methods to ensure that the optimized quantization parameters obtained from a small calibration dataset will perform well on a larger test dataset. This is a common challenge faced by all post-training quantization methods. Instead of relying on the exact quantization error for particular input images, we aim to find a suitable quantization scheme that yields a small quantization error. To tackle this issue, in this work, we use a probabilistic formulation for the quantization problem. By introducing an MDL prior and validation procedure, the probabilistic model demonstrates strong ability when learning from a small calibration dataset.

C. Modeling Quantization Using Noise Models

In this study, we focus on the uniform affine quantization approach [42], which uses a scale factor $S \in \mathbb{R}$ and a zero-point $Z \in \mathbb{Z}$ to map floating-point values to integers. The scale factor specifies the step size of the quantization. The zero-point is an integer, ensuring that zeros are quantized correctly. The uniform affine quantization for a floating-point variable $X \in \mathbb{R}$ is performed as follows:

$$\hat{X} = S\left[\mathrm{clip}\left(\lfloor X/S \rceil + Z; 0, 2^B - 1\right) - Z\right], \tag{3}$$

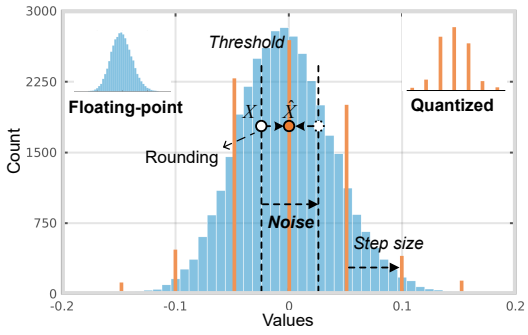

Fig. 2: The noise distribution caused by the rounding operator.

where $\lfloor \cdot \rceil$ denotes the rounding operator that rounds the variable to the nearest integer, and $\mathrm{clip}(\cdot)$ denotes the clipping operator that restricts the data to the range of $[0, 2^B - 1]$, where $B \in \mathbb{N}$ denotes the quantization bit width. $\hat{X}$ represents the quantized version for approximating the floating-point variable X. Through quantization, the quantization grid limits the range $R = [\alpha, \beta]$ with $\alpha = -SZ$ and $\beta = S(2^B - 1 - Z)$, which is also known as the quantization range.

In the quantization operation, the quantization errors resulting from the rounding operator can be treated as additive random noise. As shown in Fig. 2, when floating-point values that lie between two integer grids are rounded to the nearest integer, the noise is distributed in a finite interval with a width corresponding to the quantization step size.

The effect of uniform quantization can be approximately modeled by additive noise that is uniformly distributed within the range of $[-S/2, S/2]$ and is uncorrelated with the input signal [43]. Therefore, we can model the quantization operation as follows:

$$\hat{X} = X + \varepsilon; \quad \varepsilon \sim \mathcal{U}\left(-\frac{1}{2}S, \frac{1}{2}S\right), \tag{4}$$

where ε denotes random quantization noise. Modeling quantization with uniform noise is a popular strategy in end-to-end trained compression [44]–[46]. However, to the best of our knowledge, this work first exploits it for post-training quantization.

We assume that the quantization noise within neural networks is independent and identically distributed (i.i.d)[1] similar to [45], [46]. We denote the quantization noise as a tuple by $\boldsymbol{\varepsilon} = (\boldsymbol{\varepsilon}_l)_{l=1}^{L} \in \mathbb{R}^K$ and each vector $\boldsymbol{\varepsilon}_l = [\boldsymbol{\varepsilon}_l^{\mathrm{a}} \ \boldsymbol{\varepsilon}_l^{\mathrm{w}}] \in \mathbb{R}^{K_l}$, where l represents the index of layers and $K = \sum_{l=0}^{L} K_l$ denotes the number of noise values of all layers[2]. $\boldsymbol{\varepsilon}_l^{\mathrm{a}} \in \mathbb{R}^{A_l}$ and $\boldsymbol{\varepsilon}_l^{\mathrm{w}} \in \mathbb{R}^{J_l}$ represent the quantization noise originating from activation quantization and weight quantization, which maintain the same shapes as the weights and activations in the l-th layer. In this research, we treat quantization step sizes as

[1]We remark that for multi-layer neural networks, there are situations in which a statistical dependency between quantization noise samples can exist, especially considering zero values as outputs of a ReLU. However, we ignore this effect here to keep the analysis simple.

[2]We adopt an abuse of notation in which $\boldsymbol{\varepsilon}_l$ represents a quantization noise vector associated with the l-th layer (i.e., the concatenated weights and activation quantization noise components), and each vector $\boldsymbol{\varepsilon}_l$ has a different length ($\boldsymbol{\varepsilon}_l \in \mathbb{R}^{K_l}$). This notation simplifies the expressions and corresponds to common practices.

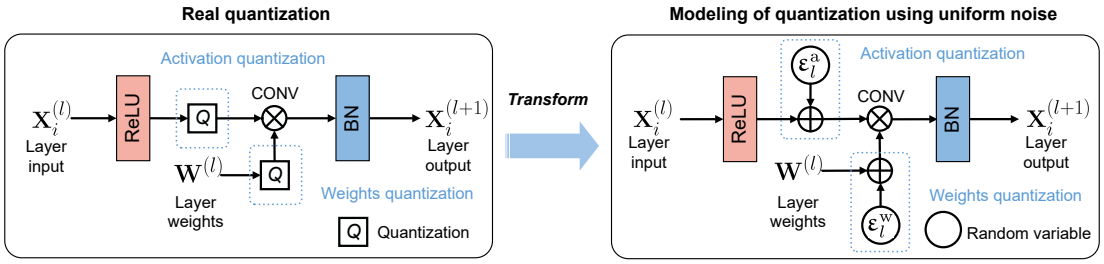

Fig. 3: The quantization operations in a neural network block are modeled by uniformly distributed random noise.

random variables. According to Eq. 4, the quantization noise depends on the quantization step sizes $\mathbf{S} = (\mathbf{S}_l)_{l=1}^{L} \in \mathbb{R}^K$ with each vector $\mathbf{S}_l \in \mathbb{R}^{K_l}$. Thus, the quantization noise in the neural network can be modeled by a uniform probability density function by

$$\begin{aligned} f_{\boldsymbol{\varepsilon}|\mathbf{S}}(\boldsymbol{\varepsilon}|\mathbf{S}) &= \prod_{l=1}^{L} f_{\boldsymbol{\varepsilon}_l|\mathbf{s}_l}(\boldsymbol{\varepsilon}_l|\mathbf{S}_l); \\ \boldsymbol{\varepsilon}_l|\mathbf{S}_l \sim \mathcal{U}(\mathcal{B}_l),\ \mathcal{B}_l &= \times_{k=1}^{K_l} \left[-[\mathbf{S}_l]_k/2, [\mathbf{S}_l]_k/2\right], \end{aligned} \tag{5}$$

where $[\mathbf{S}_l]_k$ denotes the k-th element in $\mathbf{S}_l$ and $\times$ denotes Cartesian product of intervals.

In our model, each weight and activation of the quantized layers is assigned independent quantization noise that depends on its own step size. However, in the implementation of this method, the parameter sharing is employed for quantization step sizes within each channel (e.g., per-channel quantization), and the results are obtained under this setting. As shown in Fig. 3, the quantization operations in a neural network can be modeled using additive uniform noise sources, enabling inference over the corresponding probabilistic model.

D. Bayesian Estimation of Quantization Step Sizes

Let $\mathcal{D}$ denote a dataset with N i.i.d. input-output pairs $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{N}$ with $\mathbf{X}_i \in \mathbb{R}^I$ and $\mathbf{Y}_i \in \mathbb{R}^O$, and let $\boldsymbol{\theta} = (\boldsymbol{\theta}_l)_{l=1}^{L} \in \mathbb{R}^D$ denote the parameters of a pre-trained network. Based on the uniform modeling of quantization noise, we derive the conditional distribution of the neural network outputs $\mathbf{Y} = (\mathbf{Y}_i)_{i=1}^{N} \in \mathbb{R}^{N\cdot O}$ given the input data $\mathbf{X} = (\mathbf{X}_i)_{i=1}^{N} \in \mathbb{R}^{N\cdot I}$, neural network parameters $\boldsymbol{\theta}$, and quantization step sizes $\mathbf{S} = (\mathbf{S}_l)_{l=1}^{L} \in \mathbb{R}^K$. The probabilistic relationships among different random variables are described in Fig. 4. The computation of the conditional distribution $f_{\mathbf{y}|\mathbf{s},\mathbf{x}}(\mathbf{Y}|\mathbf{S},\mathbf{X})$ requires marginalization over the hidden random variable $\boldsymbol{\varepsilon} = (\boldsymbol{\varepsilon}_l)_{l=1}^{L} \in \mathbb{R}^K$.

$$\begin{aligned} f_{\mathbf{y}|\mathbf{s},\mathbf{x}}(\mathbf{Y}|\mathbf{S},\mathbf{X}) &= \int_{\mathbb{R}^K} f_{\mathbf{y}|\boldsymbol{\varepsilon},\mathbf{x}}(\mathbf{Y}|\boldsymbol{\varepsilon},\mathbf{X}) f_{\boldsymbol{\varepsilon}|\mathbf{s}}(\boldsymbol{\varepsilon}|\mathbf{S}) \mathrm{d}\boldsymbol{\varepsilon} \\ &= \prod_{i=1}^{N}\prod_{l=1}^{L} \int_{\mathcal{B}_l} f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\boldsymbol{\varepsilon}_l,\mathbf{X}_i) f_{\boldsymbol{\varepsilon}_l|\mathbf{s}_l}(\boldsymbol{\varepsilon}_l|\mathbf{S}_l) \mathrm{d}\boldsymbol{\varepsilon}_l. \end{aligned} \tag{6}$$

In Eq. 6, we apply a factorization of the likelihood taking into account the parent-child relationships of the nodes of the Bayesian network in Fig. 4, e.g., $f_{\mathbf{y}|\boldsymbol{\varepsilon},\mathbf{x}}(\mathbf{Y}|\boldsymbol{\varepsilon},\mathbf{X}) = f_{\mathbf{y}|\boldsymbol{\varepsilon},\mathbf{s},\mathbf{x}}(\mathbf{Y}|\boldsymbol{\varepsilon},\mathbf{S},\mathbf{X})$. In the last line, we exploit the conditional independence of the input-output pairs and the quantization noise, which allows for the interchange between integration and products. In addition, the pre-trained neural network parameter $\boldsymbol{\theta}$ is omitted to simplify the notations as it is always constant in this paper.

Rather than treating the quantization step sizes as constants as in related works, this work considers the quantization step sizes as random variables with known prior distributions. Thus, we can obtain the posterior of the quantization step sizes by employing Bayes' rule as

$$f_{\mathbf{s}|\mathbf{y},\mathbf{x}}(\mathbf{S}|\mathbf{Y},\mathbf{X}) = \frac{f_{\mathbf{y}|\mathbf{s},\mathbf{x}}(\mathbf{Y}|\mathbf{S},\mathbf{X}) f_{\mathbf{s}}(\mathbf{S})}{f_{\mathbf{y}|\mathbf{x}}(\mathbf{Y}|\mathbf{X})}. \tag{7}$$

Following the Minimum Description Length (MDL) principle [47], we introduce a special prior probability for quantization step sizes, which penalizes high quantization bit widths and favors low quantization bit widths. Additionally, because the largest step size will not exceed the maximum value of weights or activations, and the smallest step size is always non-negative, we limit the step size random variables to a bounded multidimensional space $\mathcal{S} = \{(\mathbf{S}_l)_{l=1}^{L} \in \mathbb{R}^K : [\mathbf{S}_l^{\min}]_k \leq [\mathbf{S}_l]_k \leq [\mathbf{S}_l^{\max}]_k$ for all $l \in \{1,...,L\}$ and $k \in \{1,...,K_l\}\}$, where $\mathbf{S}_l^{\min} \in \mathbb{R}^{K_l}$ and $\mathbf{S}_l^{\max} \in \mathbb{R}^{K_l}$ represent the minimum thresholds and maximum thresholds of quantization step sizes in the l-th layer, respectively. Following a layer-wise optimization strategy, we assume that the quantization step sizes are statistically independent random variables, which can be modeled using the following prior distribution.

$$\begin{aligned} f_{\mathbf{s}}(\mathbf{S}) &= \prod_{l=1}^{L} f_{\mathbf{s}_l}(\mathbf{S}_l); \\ f_{[\mathbf{s}_l]_k}([\mathbf{S}_l]_k) &= \begin{cases} [\mathbf{C}_l]_k \cdot \exp\left(-\gamma \log_2 \dfrac{[\mathbf{R}_l]_k}{[\mathbf{S}_l]_k}\right), & [\mathbf{S}_l]_k \in \mathcal{S}_k; \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \tag{8}$$

where $\mathcal{S}_k = [[\mathbf{S}_l^{\min}]_k, [\mathbf{S}_l^{\max}]_k]$, with $k \in \{1,...,K_l\}$. $\mathbf{R}_l \in \mathbb{R}^{K_l}$ represents the per-channel data ranges of the l-th layer, where parameter sharing is also employed within each channel. $\gamma \in \mathbb{R}$ is a parameter to adjust the level of penalization. $\mathbf{C}_l \in \mathbb{R}^{K_l}$ is a vector of normalization constants of probability distributions. The term $\log_2\left([\mathbf{R}_l]_k/[\mathbf{S}_l]_k\right)$ is used to calculate the quantization bit width, where the ceiling operation is removed to facilitate gradient computation. This prior makes random step size variables lean towards large values corresponding to low bit widths.

However, computing the exact posterior is intractable because the denominator requires integrating over continuous random variables. Approximate Bayesian inference methods are commonly used as alternatives to Bayesian inference [48].

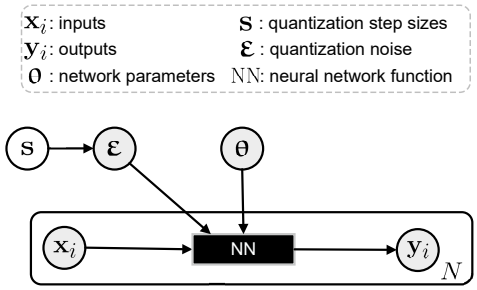

Fig. 4: The probabilistic programming graphical model[3].

In this research, we employ a probabilistic programming technique in combination with Monte Carlo methods to obtain approximate Bayesian MAP inference. According to Eq. 7, as the denominator is independent of the quantization step sizes $\mathbf{S}$, maximizing the posterior probability of $\mathbf{S}$ is equivalent to maximizing the joint distribution of quantization step sizes and network outputs $\mathbf{Y}$ given network inputs $\mathbf{X}$. With probabilistic programming, the parameters of distributions (i.e., quantization step sizes), can be set as optimizable parameters, which enables maximization using gradient-based methods. Thus, the probabilistic optimization program for the posterior of step sizes can be performed as

$$\begin{aligned}
&\arg\max_{\mathbf{S}\in\mathcal{S}} f_{\mathbf{s}|\mathbf{y},\mathbf{x}}(\mathbf{S}|\mathbf{Y},\mathbf{X}) \\
&\quad = \arg\max_{\mathbf{S}\in\mathcal{S}} f_{\mathbf{y}|\mathbf{s},\mathbf{x}}(\mathbf{Y}|\mathbf{S},\mathbf{X}) f_{\mathbf{s}}(\mathbf{S}) \\
&\quad = \arg\max_{\mathbf{S}\in\mathcal{S}} \prod_{i=1}^{N}\prod_{l=1}^{L} \int_{\mathcal{B}_l} f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\boldsymbol{\varepsilon}_l,\mathbf{X}_i)\cdot \\
&\qquad\qquad f_{\boldsymbol{\varepsilon}_l|\mathbf{s}_l}(\boldsymbol{\varepsilon}_l|\mathbf{S}_l) f_{\mathbf{s}_l}(\mathbf{S}_l)\mathrm{d}\boldsymbol{\varepsilon}_l.
\end{aligned} \tag{9}$$

In the last line of Eq. 9, the marginal likelihood in Eq. 6 is taken into account.

The implementation of probabilistic programming techniques to optimize Eq. 9 will be explained in Section IV.

E. Percentile-based Clipping Method

In the quantization operation, any value outside the quantization range $[\alpha, \beta]$ will be clipped to α and β, which incurs clipping errors. Although the clipping error can be entirely avoided by choosing a sufficiently large quantization range, this is not beneficial in terms of rounding errors since strong outliers may cause excessive rounding errors [41]. Quantization range searching has been one effective way to alleviate the issue of strong outliers. This study introduces a novel quantization range searching method.

Empirically, the strong outliers typically deviate far from the center of the distribution and account for a small proportion of the overall data. The proposed method is to distinguish strong outliers by utilizing the percentile of an approximate distribution. Assume activations, denoted by a random variable x, approximately follow a Gaussian distribution with mean μ and standard deviation σ (this assumption has been adopted in many quantization studies [10], [17], [26]). The probability of an observed value X that exceeds a given bound τ can be calculated by

$$\Pr(X > \tau) = \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\frac{\tau-\mu}{\sqrt{2}\sigma}\right), \tag{10}$$

where $\mathrm{erf}(\cdot)$ is the error function.

If the percentage of the random variable x exceeding a bound τ is very low, we can utilize this bound to determine the clipping thresholds of the quantization ranges. Following Eq. 10, the bound (percentile) τ with respect to a given probability $P = \Pr(X > \tau)$ can be calculated as follows:

$$\tau = \sqrt{2}\sigma\mathrm{erf}^{-1}(1-2P) + \mu, \tag{11}$$

where $\mathrm{erf}^{-1}(\cdot)$ represents the inverse error function. In fact, the real mean and the real standard deviation are often unknown. However, we can estimate them using the sampled mean $\hat{\mu}$ and the sampled standard deviation $\hat{\sigma}$ calculated from observed data, which can be obtained from the calibration dataset. Thus, the bound (percentile) τ can be estimated by

$$\hat{\tau} = \sqrt{2}\hat{\sigma}\mathrm{erf}^{-1}(1-2P) + \hat{\mu}. \tag{12}$$

The obtained $\hat{\tau}$ value can be utilized to determine an upper clipping threshold by $\beta = \hat{\tau}$. Based on the symmetry property of the Gaussian distribution, the lower clipping threshold is determined by $\alpha = 2\hat{\mu} - \hat{\tau}$. For one neural network, all layers can utilize a unified probability P to determine the clipping thresholds of all layers because the specific clipping thresholds for each layer are adaptively determined based on the distributions of activations.

To select the appropriate probability P, we design an evaluation criterion to assess its performance. This criterion is the cross-entropy loss between the quantized outputs and unquantized outputs on the final layer. First, we generate a list of probability candidates spanning a wide range. Then, we quantize the neural network using clipping thresholds calculated from each probability, and measure the cross-entropy loss on the final layer. Finally, the probability candidate that yields the smallest loss value is selected to determine the clipping thresholds for quantization. Different from local quantization error-based clipping methods [10], [13], [27], which is an approximate way to judge the final performance degradation, the proposed clipping method directly measures the final performance degradation so that it is more accurate compared to local error-based clipping methods.

Based on standard practice in the quantization field [10], [41], we also clip the weights, despite the fact that the weights are constant and known. Although the preferred method of choice would be to obtain clipping thresholds using empirical percentiles, we still use a Gaussian approximation because the empirical percentiles are not reliable due to the small number of weights per channel.

The proposed clipping method is performed only once before the quantization optimization. For activation clipping, to avoid introducing extra computational cost, we calculate the average clipping thresholds on the calibration dataset and fix them during the inference stage.

[3]White circles represent latent random variables. Gray circles indicate observed random variables. Arrows show dependency and forward data flow. Rounded rectangles represent N distributions.

IV. Implementation using A Probabilistic Programming Language

In this section, we will explain the implementation of the inference model with Probabilistic Programming Languages (PPLs). The PPLs allow for directly expressing probabilistic models in programming code, which makes PPLs ideal for implementing our proposed quantization method. However, because not all steps of the probabilistic inference are directly supported by PPLs, we make a few adaptions so that the probabilistic inference can be easily implemented. First, Monte Carlo integration and a reparameterization trick are employed to facilitate computation. Next, we introduce an approximation for the neural network function and establish conditional dependency using PPLs. Finally, we give a practical optimization objective. In this work, two probabilistic programming languages are employed: TensorFlow Probability (TFP) [49] and Pyro [21].

A. Monte Carlo Estimate and Reparameterization

Considering the high dimensions and the continuity of random variables, it is intractable to calculate the marginal likelihood explicitly. As a practical approximation, Monte Carlo integration is often employed. Furthermore, to optimize Eq. 9 using gradient-based methods, it is essential that the optimization objective is in a differentiable form, allowing for obtainable gradients with respect to parameters. In this study, the network quantization noise is modeled using uniform distributions. However, the parameters of the uniform distribution cannot be directly optimized due to the constant (non-parametric) bounds provided by PPLs, e.g., the uniform distribution in TFP. To facilitate the optimization of the quantization step sizes, a commonly used gradient approximation strategy, known as the reparameterization trick [50], [51], is employed to reparameterize quantization noise.

Let $\boldsymbol{\nu}$ represent a vector of independent and identically distributed (i.i.d.) random variables following uniform distributions where each element is distributed over the interval $[-1/2, 1/2]$, denoted by a tuple as $\boldsymbol{\nu} = (\boldsymbol{\nu}_l)_{l=1}^{L} \in \mathbb{R}^K$ with each vector $\boldsymbol{\nu}_l \in \mathbb{R}^{K_l}$. The reparameterization trick performs as

$$\boldsymbol{\varepsilon} = \mathbf{S} \odot \boldsymbol{\nu}, \tag{13}$$

where $\odot$ denotes the Schur product (i.e., elementwise product). The reparameterization trick is an effective strategy for making the Monte Carlo estimate of the expectation differentiable with respect to parameters of distributions, which has wide applications in recent advances in machine learning [52].

Through the use of the reparameterization trick and Monte Carlo integration, the optimization objective of Eq. 9 can be transformed into

$$\begin{aligned}\arg\max_{\mathbf{S}\in\mathcal{S}} \prod_{i=1}^{N}\prod_{l=1}^{L} \int_{\Omega_{\boldsymbol{\nu}_l}} & f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\mathbf{S}_l \odot \boldsymbol{\nu}_l, \mathbf{X}_i) \cdot \\ & f_{\boldsymbol{\nu}_l}(\boldsymbol{\nu}_l) f_{\mathbf{s}_l}(\mathbf{S}_l) \mathrm{d}\boldsymbol{\nu}_l \\ = \arg\max_{\mathbf{S}\in\mathcal{S}} \prod_{i=1}^{N}\prod_{l=1}^{L}\sum_{j=1}^{M} & f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\mathbf{S}_l \odot \boldsymbol{\nu}_l^{(j)}, \mathbf{X}_i) f_{\mathbf{s}_l}(\mathbf{S}_l),\end{aligned} \tag{14}$$

```
model = tfp.distributions.JointDistributionSequential([
  # Uniform distribution in the range of [-0.5, 0.5].
  tfp.distributions.Uniform(low=-0.5*tf.ones_like(W_l),
                            high=0.5*tf.ones_like(W_l))  # n
  # Modeling the neural network using a Gaussian function.
  lambda n: tfp.distributions.Independent(
    # Conditional dependency and reparameterization.
    tfp.distributions.Normal(loc=NN(images, tf.multiply(S_l, n)),
                             scale=0.01), reinterpreted_batch_ndims=0)])
```

Fig. 5: The implementation of conditional dependence for weight quantization of a single layer using TFP.

where $\Omega_{\boldsymbol{\nu}_l} = \left[-\frac{1}{2}, \frac{1}{2}\right]^{K_l}$ and $\boldsymbol{\nu}_l^{(j)}$ denotes the j-th Monte Carlo realization of $\boldsymbol{\nu}_l$ that is uniformly drawn from $\Omega_{\boldsymbol{\nu}_l}$.

B. Modeling Neural Network Function

Given input data $\mathbf{X}$ and quantization noise values $\boldsymbol{\varepsilon}$, the neural network function can be modeled in the probabilistic programming language using a Dirac delta distribution as

$$f_{\mathbf{y}|\boldsymbol{\varepsilon},\mathbf{x}}(\mathbf{Y}|\mathbf{S} \odot \boldsymbol{\nu}, \mathbf{X}) = \delta(\mathbf{Y} - \mathrm{NN}(\mathbf{X}; \boldsymbol{\theta}, \mathbf{S} \odot \boldsymbol{\nu})), \tag{15}$$

where $\mathrm{NN}(\cdot)$ denotes a quantized neural network function given pre-trained network parameters $\boldsymbol{\theta}$ and quantization noise values $\boldsymbol{\varepsilon}$.

However, TFP does not have a direct way to specify the Dirac delta distribution, nor does it support non-bijective transforms of random variables (in this case, the neural network function). In addition, the density of the Dirac delta distribution is not continuous, which is not beneficial for gradient-based backpropagation. In the definition of the TFP models, a pragmatic solution is then to approximate the Dirac delta distribution using a Gaussian function with small variance $\eta\mathbf{I}_O$, where $\mathbf{I}_O$ is the $O \times O$ identity matrix.

$$\begin{aligned} f_{\mathbf{y}|\boldsymbol{\varepsilon},\mathbf{x}}(\mathbf{Y}|\mathbf{S} \odot \boldsymbol{\nu}, \mathbf{X}) &= \prod_{i=1}^{N}\prod_{l=1}^{L} f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\mathbf{S}_l \odot \boldsymbol{\nu}_l, \mathbf{X}_i); \\ \mathbf{Y}_i|\mathbf{S}_l \odot \boldsymbol{\nu}_l, \mathbf{X}_i &\sim \mathcal{N}\left(\mathrm{NN}(\mathbf{X}_i; \boldsymbol{\theta}, \mathbf{S}_l \odot \boldsymbol{\nu}_l), \eta\mathbf{I}_O\right), \end{aligned} \tag{16}$$

where the outputs of the network function are set as the mean of the Gaussian distribution, and η is a small value.

Specifically, applying Bayesian inference requires access to an observation $\mathbf{Y}_i$ originating from a neural network quantizer that uses the desired (i.e., to be estimated) quantization step sizes. Because such observation is not available, instead, we utilize $\mathbf{Y}_i = \mathrm{NN}(\mathbf{X}_i; \boldsymbol{\theta}, \mathbf{0})$ as a proxy for the observed output values. In Section A of the supplementary material, we show that this choice corresponds to a MAP-like estimator that minimizes the expected zero-one quantization error. In the next subsection, we will discuss the implementation of conditional dependence in Eq. 16.

C. Establishment of Conditional Dependency

To obtain the conditional distribution of quantized outputs in Eq. 16, we need to establish a conditional dependency between the distributions of neural network outputs and the distributions of quantization noise. To achieve this, we utilize

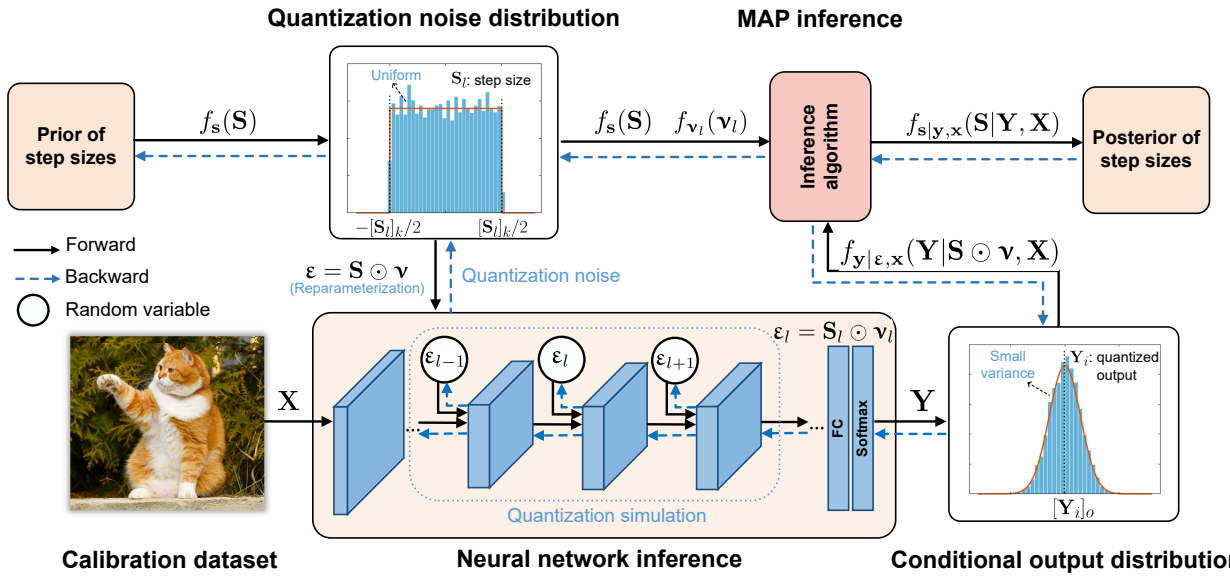

Fig. 6: The illustration of the probabilistic programming quantization (PPQ) method. The quantization step sizes are treated as random variables and a special prior probability is introduced to penalize high quantization bit widths. The quantization noise is modeled by uniform distributions, whose values are imported into deep neural networks to simulate real quantization. To model the conditional neural network output distribution, a conditional Gaussian distribution with a small variance is used. By maximizing the the joint probability of quantization step sizes and neural network outputs, maximum posterior estimates of the step sizes can be obtained. The quantization parameters of step sizes are optimized using gradient descent algorithms.

a function provided by the TFP, *JointDistributionSequential* [53], to create this relationship by specifying a sequence of conditional distributions that follow multiplication rules.

Fig. 5 illustrates an implementation for weight quantization of a single layer using the TFP probabilistic programming language. The approximate distribution of neural networks, conditioned on the quantization noise distributions, is expressed by means of a *lambda* expression. The function *Independent* is used to construct a joint distribution as the product of multiple distributions on statistically independent random variables (each of the distributions having different calibration images and unified noise samples as input)[4].

Fig. 6 illustrates the overall architecture of the proposed PPQ method. The optimization details will be discussed in the following subsection.

D. Optimization Objective

To maximize the posterior probability of quantization step sizes using the probabilistic programming technique, the gradient-based methods are required. Considering the global optimum is difficult to find in reasonable computation time due to the high computational and memory complexity, we employ a practical layer-by-layer optimization strategy to optimize the objective Eq. 9.

For convenience, we maximize the logarithm of the layer-wise optimization objective function in order to obtain better-scaled gradients for optimized distribution parameters. Given a calibration dataset with N input-output pairs, the layer-wise optimization objective becomes as follows:

$$\arg\max_{\mathbf{S}_l \in \mathcal{S}_l} \sum_{i=1}^{N} \log \sum_{j=1}^{M} f_{\mathbf{y}_i|\boldsymbol{\varepsilon}_l,\mathbf{x}_i}(\mathbf{Y}_i|\mathbf{S}_l \odot \boldsymbol{\nu}_l^{(j)}, \mathbf{X}_i) + \log f_{\mathbf{s}_l}(\mathbf{S}_l), \tag{17}$$

where $\mathcal{S}_l = \times_{k=1}^{K_l} \left[[\mathbf{S}_l^{\min}]_k, [\mathbf{S}_l^{\max}]_k\right]$. To solve this optimization problem, we employ the projected gradient descent method [54], which performs the gradient projection to ensure variables $\mathbf{S}_l$ within the constrained range $\mathcal{S}_l$ during the optimization process.

To tackle the issue of overfitting when optimized with a small dataset, we introduce a validation procedure to select quantization parameters. In network training, the best model checkpointing is commonly used to save the best weights through validating on a validation dataset [55], [56]. Inspired by this, we split the calibration dataset into two equal parts; one part is used for optimizing parameters and another part is served as a validation dataset. Similar to the percentile-based clipping method, the validation procedure measures the cross-entropy loss between the unquantized outputs and the quantized outputs of the final layer. The quantization loss on the final layer usually serves as an effective metric for evaluating quantization performance [12], [13], [29].

During the optimization process, we perform the validation every two iterations. This number was determined experimentally to balance the performance of the validation procedure versus the computational overhead. At each validation step, we use the temporal values of the optimized step sizes to quantize neural networks and then calculate the quantization loss of the final layer. The final quantization parameters are determined by the step size values that yield the smallest loss during the

[4]For more details about these functions, the reader can refer to the official documents of the TFP on https://www.tensorflow.org/probability

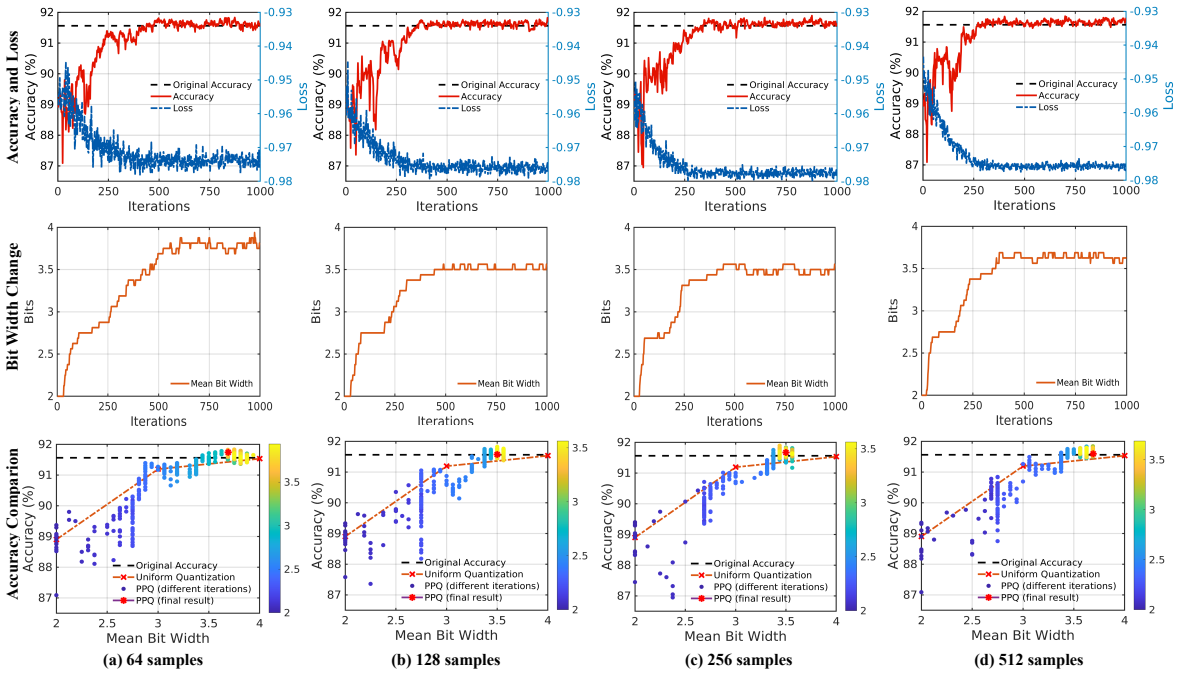

Fig. 7: Optimizing weight quantization for a single layer in ResNet20 as a function of various calibration dataset sizes. The proposed method achieves nearly the same accuracy as the original with fewer bits compared to the baseline method.

optimization process. The execution process is demonstrated in Fig. 1.

Additionally, early stopping has served as a good strategy to prevent overfiting in Bayesian optimization research [57], [58]. In our method, an early stopping method is employed to stop further optimization when there is no better solution than the smallest loss value from the successive one hundred validation steps. Although the validation procedure cannot absolutely avoid overfitting when relying on a small dataset, our experiments show that the validation procedure effectively helps to improve the optimization performance and enables us to select the good parameters when learning from a small calibration dataset, which will be demonstrated in Section V.

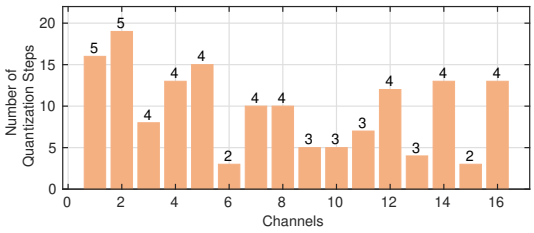

Fig. 8: The optimized number of quantization steps for the different channels of one layer's weights in ResNet20. The annotated number above each bar are the required bit width.

V. Experiments

In this section, we conduct a comprehensive evaluation to verify the performance of the proposed post-training quantization method. We examine the effectiveness of the proposed quantization method and assess the performance of the proposed clipping method. Next, we compare our method with existing post-training quantization methods. Finally, we evaluate the generalization ability by optimizing with different calibration sources and testing on diverse datasets.

A. Experimental Setup

The proposed PPQ method requires an unlabeled calibration dataset for the optimization process. The calibration datasets are randomly selected from the training dataset of ImageNet [59] and CIFAR10 [60]. Following the common practice [29], [33], we quantize the first layer and the last layer to 8-bit as these layers typically have a greater impact on accuracy degradation. We employ a per-channel quantization strategy for the numerical results. For weight quantization, we apply the bias correction method proposed in [26] on the top of the proposed method to further improve quantization performance.

The PPQ method optimizes each layer over 1000 iterations using the Adam optimizer [61]. In our experiments, we set the sample number M to 16 except in the experiments incorporating with the bias correction method where M is set to 2. Unless stated otherwise, the minimum thresholds $\mathbf{S}_l^{\min}$ and maximum thresholds $\mathbf{S}_l^{\max}$ for prior ranges are set to the step sizes corresponding to naive uniform affine quantization with 8-bit and 3-bit precision, respectively.

We evaluate the quantization performance of the proposed method on computer vision tasks for classification and object detection with different neural network models, including ResNet [62], MobileNetV2 [63], RetinaNet [64], and vision transformers [65], [66]. All experiments employ the TensorFlow deep learning framework, except for the vision transformer experiments, which are implemented using the PyTorch framework.

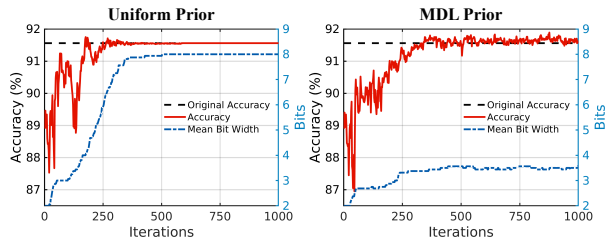

Fig. 9: Variation of accuracy and bit width with a uniform prior (left graph) and a MDL prior (right graph). Using MDL prior significantly reduces the bit width while retaining the accuracy on the validation dataset.

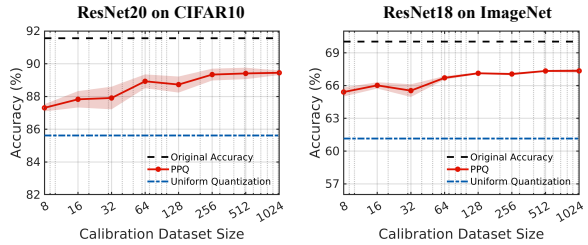

Fig. 10: Quantization performance with various calibration dataset sizes on CIFAR10 and ImageNet datasets.

B. Quantization with Probabilistic Programming

We assess the performance of the proposed PPQ method on different calibration dataset sizes, ranging from 64 samples to 512 samples. The ResNet20 is utilized as evaluation. We record the top-1 accuracy, loss, and quantization bit width changes during optimization. In this case, we set $\mathbf{S}_l^{\max}$ to step sizes corresponding to 2-bit naive uniform affine quantization. The results are shown in Fig. 7.

The first row shows the variations in accuracy and loss (i.e., negative values of the optimization objective of Eq. 17). Due to the random sampling of Monte Carlo integration, the loss values fluctuate continuously. They gradually decrease to relatively stable states that fluctuate around certain values. All four calibration sizes exhibit significant improvement in accuracy during the optimization process. However, due to the fluctuation of accuracy, the final accuracy achieved is not necessarily the highest point during the optimization process. Nevertheless, the validation procedure can help avoid suboptimal results by selecting quantization parameters.

Because the step size of each channel is optimized independently, the bit width varies across channels, which results in a mixed-precision quantization scheme. As shown in Fig. 8, the number of quantization steps (the ceiling of the data range divided by quantization step sizes) varies across different channels, resulting in varying quantization bit widths in different channels. This shows that the proposed method enables adaptive adjustment of the quantization step size variables for different channels.

The second row of Fig. 7 shows the mean bit width change for a single layer during the optimization process. Throughout the optimization process, the mean bit width gradually increases until the stable state is reached.

In the third row, we perform an accuracy comparison with the naive uniform affine quantization [42]. For all calibration

TABLE I: Performance comparison of the proposed percentile-based clipping method against other clipping methods on the ImageNet dataset.

Models	ResNet18		ResNet50	
Bit width	4 bits	3 bits	4 bits	3 bits
Weight Quantization (W4A32)				
None	61.15	4.68	73.24	41.37
CFP	61.50	10.70	69.53	55.90
MSE	61.51	27.10	73.23	50.71
Ours	**66.14**	**33.96**	**73.53**	**62.19**
Activation Quantization (W32A4)				
None	65.30	39.97	73.84	45.09
CFP	66.24	50.83	73.72	55.33
MSE	68.37	63.24	74.98	**71.57**
Ours	**68.58**	**63.81**	**75.05**	70.69

TABLE II: Ablation study on the proposed PPQ quantization method applied to 4-bit weight quantization of ResNet18 on the ImageNet dataset.

Methods	Acc.(%)
Baseline	61.15
Percentile-based Clipping	66.14
PPQ (sole)	67.06
PPQ + Percentile-based Clipping	67.62
Bias correction	68.77
PPQ (combined)	**69.12**
Floating-point	70.03

dataset sizes, the proposed method manages to exceed the accuracy of 4-bit naive uniform affine quantization with a bit width of less than 4 bits. This experiment demonstrates that the PPQ method outperforms the baseline method when learning from small calibration datasets.

The accuracy corresponding to the selected parameters from the validation procedure is also reported in the third row, which is denoted by red stars. The validation procedure almost identifies the highest results. This demonstrates the effectiveness of validation based on a small calibration dataset.

To evaluate the effect of the introduced MDL prior, we compare its performance against a uniform (non-informative) prior choice. Using the same experimental setup in the previous tests, we track changes in accuracy and bit width throughout the optimization process. The results are shown in Fig. 9.

The experiment with a uniform prior exhibits a significant accuracy increase during optimization. However, using the MDL prior achieves almost the same accuracy while using significantly fewer bits.

We evaluate the performance of weight quantization on the whole ResNet20 and ResNet18 with different calibration dataset sizes, ranging from 8 samples to 1024 samples. To highlight improvements, we deliberately select relatively low quantization precision: 3-bit quantization for ResNet20 and 4-bit quantization for ResNet18. The PPQ method is controlled to obtain approximately the same bit widths. In Section B of the supplementary material, we show experiments using

TABLE III: Comparison with existing post-training quantization methods on the ImageNet validation dataset. Top-1 accuracy (%) and the accuracy degradation (in parentheses) compared with the pre-training accuracy (in respective papers) are reported. W/A denotes weights/activations. The result indicated by † is taken from [13]. 4* represents an approximate 4-bit (< 4.1-bit).

Methods	Bits (W/A)	Mixed-Precision	Calibration Samples	ResNet18	ResNet50	MobileNetV2
Baseline [42]	4/8	✗	0	61.15 (-8.88)	73.24 (-2.96)	2.39 (-69.33)
SPARQ [28]	4/8	✗	2000	67.49 (-2.27)	75.03 (-1.10)	-
ACIQ-Mix [10]	4/8	✓	0	68.30 (-1.40)	75.30 (-0.80)	-
BC [26]	4/8	✗	512	68.75 (-1.28)	75.44 (-0.76)	61.99 (-9.73)
Adaround [12]	4/8	✗	2048	68.55 (-1.13)	75.01 (-1.06)	**69.25 (-2.47)**
Bit-Split [29]	4/8	✗	1024	69.10 (-0.66)	75.58 (-0.57)	68.41 (-3.46)
GPTQ [67]	4/32	✗	1024	**69.37 (-0.39)**	**75.71 (-0.42)**	-
PPQ (ours)	4*/8	✓	256	69.17 (-0.86)	75.64 (-0.56)	68.92 (-2.80)
PPQ (ours)	4*/8	✓	64	68.97 (-1.06)	75.54 (-0.66)	68.53 (-3.19)
Baseline [42]	4/4	✗	0	56.24 (-13.79)	70.80 (-5.40)	2.06 (-69.66)
LAPQ [32]	4/4	✗	512	60.30 (-9.40)	70.00 (-6.10)	52.20 (-19.60)
ACIQ-Mix [10]	4/4	✓	-	67.00 (-2.70)	73.80 (-2.30)	-
AdaQuant [33]	4/4	✓	1000	69.60 (-2.37)	75.90 (-1.30)	34.95†(-37.54)
QDROP [34]	4/4	✗	1024	69.10 (-1.96)	75.03 (-1.97)	67.89 (-4.60)
PD-Quant [35]	4/4	✗	1024	69.23 (-1.78)	75.16 (-1.47)	68.19 (-4.43)
Bit-Split [29]	4/4	✗	1024	68.10 (-1.66)	74.20 (-1.95)	49.23 (-22.64)
CL-Calib [68]	4/4	✗	128	69.41 (-1.60)	75.38 (-1.25)	68.56 (-4.06)
PFQ [30]	4/4	✗	1024	69.56 (-1.50)	75.25 (-1.75)	**69.07 (-3.42)**
BRECQ [13]	4/4	✓	1024	69.60 (-1.48)	75.05 (-1.95)	66.57 (-5.92)
Mr.BiQ [69]	4/4	✗	1000	**69.68 (-1.40)**	75.17 (-1.83)	57.27 (-15.22)
PPQ (ours)	4*/4*	✓	256	68.32 (-1.71)	**75.03 (-1.17)**	65.70 (-6.02)
PPQ (ours)	4*/4*	✓	64	68.21 (-1.82)	74.60 (-1.60)	65.11 (-6.61)

hyperparameter γ to control the expected quantization bit width. The results are shown in Fig 10.

Both models achieve nearly identical accuracy with sample sizes ranging from 256 to 1024. ResNet18 attains the highest mean accuracy of 67.35%, surpassing the baseline by 6.20%. Even with just 8 calibration samples, both ResNet20 and ResNet18 still outperform the baseline by 1.69% and 4.26%, respectively. This experiment demonstrates the good ability of the proposed method in learning from small datasets.

C. Percentile-based Clipping Results

To evaluate the proposed percentile-based clipping method, we conduct experiments on ResNet18 and ResNet50 for weight and activation quantization. The calibration dataset includes 256 samples. We compare the performance of low-precision 4-bit and 3-bit quantization with the Mean Squared Error (MSE)-based clipping method [27] and the Coarse-to-Fine Pre-processing (CFP) clipping method [70]. Some recent clipping methods [71]–[73] address strong outliers in per-token quantization for large language models (LLMs), which are not viable alternatives to the proposed quantization method. The search range for the hyperparameter P spans from 10^{-2} to 10^{-13} with 50 evenly spaced samples at each order of magnitude. The results are summarized in table I.

In terms of weight quantization, the proposed clipping method achieves the best performance. It demonstrates a 4.99% improvement in 4-bit quantization of ResNet18, surpassing the MSE-based method and the CFP method by 4.63% and 4.64%, respectively. Regarding activation quantization, the proposed clipping method still shows better performance, except for the case of 3-bit quantization on ResNet50. Overall, the proposed clipping method performs better compared to the MSE-based and CFP clipping methods.

D. Ablation Study

Except for quantization parameter optimization and quantization range searching, bias correction also plays a significant role in post-training quantization. Similar previous works [12], [32], we incorporate an existing bias correction method [26] to further enhance performance. The bias correction is only applied for weight quantization. To evaluate the impact of bias correction, we conduct an ablation study on ResNet18 models for 4-bit weight quantization using a calibration set of 256 samples. The results are summarized in table II.

Without additional methods, the sole PPQ method outperforms the baseline method by 6.18%. After combining the clipping method, the accuracy further increases by 0.56%. On the other hand, the bias correction method demonstrates a significant improvement. By combining the bias correction method, the PPQ method achieves an accuracy of 69.12%, with a only 0.91% accuracy degradation compared to the original accuracy.

E. Comparison to State-of-The-Art Methods

We compare the proposed method to existing post-training quantization methods. The results are summarized in Table

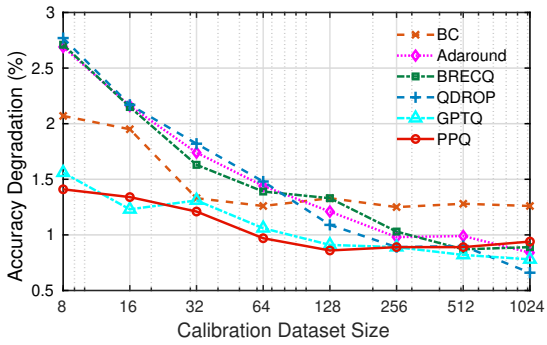

Fig. 11: Accuracy degradation across different calibration dataset sizes for 4-bit weight quantization of ResNet18 on the ImageNet dataset.

III. The comparison focuses on low precision 4-bit quantization with different models, including ResNet18, ResNet50, and MobileNetV2. For weight quantization comparisons, we quantize activations using naive uniform affine quantization. To account for the difference in the accuracy of pre-trained models, we also provide the actual accuracy degradation compared to the original accuracy reported in the relevant literature. The table includes details of the calibration dataset sizes. Our method is conducted on calibration datasets with 256 samples and 64 samples, respectively.

Compared to existing methods, the PPQ utilizes the minimal calibration dataset thanks to the Bayesian framework and validation procedure. For weight quantization, the PPQ with 256 calibration samples outperforms the Bit-Split method on MobileNetV2 by 0.66%. Besides, the PPQ achieves comparable performance with the GPTQ method on ResNet50 with smaller calibration datasets. In the case of 4-bit weight and activation quantization, the PPQ achieves the highest performance on ResNet50, with only a 1.17% accuracy degradation. With fewer 64 samples, it still shows comparable performance to other methods. Notably, the CL-Calib method also achieves a strong performance with 128 samples by leveraging contrastive learning, which incorporates multiple data augmentation operations. This experiment demonstrates that the PPQ method achieves nearly state-of-the-art performance when using a minimal calibration dataset.

Furthermore, we compare the quantization performance of the proposed method with several existing PTQ methods across various calibration dataset sizes, ranging from 8 to 1024 samples. This experiment is conducted on ResNet18 with 4-bit weight quantization. The Top-1 accuracy degradation compared to the pre-training accuracy in the respective papers is reported. The experimental results are illustrated in Fig. 11.

The proposed method exhibits the smallest accuracy variation across different calibration dataset sizes. While the QDROP and GPTQ methods achieve superior performance with large calibration datasets, e.g., 512 samples and 1024 samples, they underperform the proposed PPQ method when the calibration samples are fewer than 256. Notably, with only 8 calibration samples, the PPQ method maintains an accuracy degradation of just 1.41%, surpassing the QDROP and GPTQ methods by 1.35% and 0.15%, respectively. These results demonstrate the superior performance of the PPQ method in scenarios with small calibration datasets.

TABLE IV: The experiment on object detection using RetinaNet model for 4-bit weight quantization and 8-bit activations quantization. The bounding box average precision (AP) with different intersection over union (IoU) on the COCO2017 validation dataset is reported.

Methods	Bits (W/A)	$AP_{0.5:0.95}$	$AP_{0.5}$	$AP_{0.75}$
Baseline [42]	4/8	33.1 (-1.5)	51.7 (-1.9)	35.2 (-1.7)
BC [26]	4/8	0.0 (-34.6)	0.1 (-53.5)	0.0 (-36.9)
ACIQ [10]	4/8	34.0 (-1.6)	**54.3 (-1.2)**	36.3 (-2.0)
Bit-Split [29]	4/8	34.4 (-1.2)	54.2 (-1.3)	36.5 (-1.8)
PPQ (ours)	4*/8	**33.7 (-0.9)**	**52.4 (-1.2)**	**35.8 (-1.1)**

TABLE V: The experiment on vision transformers for 4-bit weight and activation quantization. Top-1 accuracy on the ImageNet validation dataset is reported.

Model	ViT-S	ViT-B	DeiT-T	DeiT-S	DeiT-B
Floating-point	81.39	84.54	72.21	79.85	81.8
PTQ4ViT [74]	42.57	30.69	36.96	34.08	64.39
APQ-ViT [75]	47.95	41.41	47.94	43.55	67.48
RepQ-ViT [76]	65.05	68.48	57.43	69.03	75.61
LRP-QViT [77]	70.81	75.37	61.24	72.43	78.13
ERQ [78]	71.61	78.65	61.79	**74.35**	**79.18**
PPQ (ours)	**71.95**	**78.80**	**61.86**	73.25	79.01

F. Object Detection Experiments

We conduct experiments for weight quantization on the object detection model of RetinaNet [64], which utilizes ResNet50 as its backbone. The experiment uses the Keras implementation of RetinaNet[5]. Activations are quantized to 8-bit by the naive uniform affine quantization. The calibration dataset includes 256 images randomly selected from the COCO2017 training dataset. We evaluate the accuracy on the COCO2017 validation dataset. To account for differences in pre-trained models, we also provide the accuracy degradation (in parentheses) compared with the pre-training accuracy in the respective papers. The results are shown in Table IV.

In the table, the PPQ method only experiences a degradation of 0.9%-1.2%. This outperforms the Bit-Split and ACIQ methods by up to 0.7% and up to 0.9%, respectively. This experiment demonstrates that the proposed method can achieve good performance on object detection models as well.

G. Vision Transformer Experiments

To demonstrate the generalization of the proposed method across various models, we evaluate its performance on vision transformers for the classification task. The experiment focuses on 4-bit weight and activation quantization on different models, including ViT [65] and DeiT [66] models. This experiment is based on the open-source project of RepQ-ViT [76]. For a fair comparison, the quantization settings are aligned with RepQ-ViT, which employs channel-wise weight quantization and layer-wise activation quantization. Similar to ERQ [78],

[5]https://github.com/fizyr/keras-retinanet/tree/main

TABLE VI: The experiment of using various calibration data sources for 4-bit weight quantization of ResNet18. Top-1 accuracy on the ImageNet validation dataset is reported.

Data sources	BC [26]	Bit-Split [29]	BRECQ [13]	PPQ (ours)
ImageNet	68.77 (-1.26)	**69.10 (-0.70)**	70.05 (-1.03)	69.12 (-0.91)
VOC2012	68.67 (-1.36)	68.48 (-1.32)	70.06 (-1.02)	**69.17 (-0.86)**
COCO2017	68.77 (-1.26)	68.53 (-1.27)	69.97 (-1.11)	**69.02 (-1.01)**
DeepInv	68.61 (-1.42)	68.36 (-1.44)	69.82 (-1.26)	**68.90 (-1.13)**
CIFAR10	67.76 (-2.27)	62.53 (-7.27)	66.38 (-4.70)	**68.66 (-1.37)**
Gaussian noise	**66.86 (-3.17)**	9.73 (-60.07)	58.22 (-12.86)	62.94 (-7.09)

TABLE VII: The experiment of validation on diverse datasets using ResNet18 with parameters from the ImageNet dataset. Top-1 accuracy on diverse datasets is reported.

Methods	ImageNet	VOC2012	COCO2017	DeepInv
Baseline [42]	72.26	50.93 (-21.33)	52.39 (-19.87)	58.41 (-13.85)
LAPQ [32]	73.18	50.64 (-22.72)	53.54 (-19.84)	71.51 (-1.67)
BC [26]	89.03	77.18 (-11.85)	78.84 (-10.19)	84.47 (-4.16)
Bit-Split [29]	89.95	78.67 (-11.28)	80.49 (-9.49)	**90.56 (+0.61)**
BRECQ [13]	**90.69**	**79.85 (-10.84)**	**81.76 (-8.93)**	90.44 (-0.25)
PPQ (ours)	90.07	79.08 (-10.99)	81.08 (-8.99)	86.98 (-3.09)

we apply additional rounding refinement [12] after the MAP inference optimization to further enhance performance. The calibration dataset includes 256 samples randomly selected from the ImageNet training dataset. The results are summarized in Table V.

In the table, the proposed method achieves good performance on all vision transformers. Compared to the baseline RepQ-ViT, it demonstrates a significant improvement. While the ERQ method achieves the best performance on the DeiT-S and DeiT-B models by reducing errors through dual uniform quantization and weight readjustment, the proposed PPQ method outperforms it on the other three models by 0.07%-0.34%. This experiment highlights the effectiveness and scalability of the PPQ method on larger vision transformer architectures.

H. Generalization Experiments

To assess the generalization ability of the PPQ method, we evaluate its performance of optimizing with various calibration data sources. This experiment is valuable in determining whether other data sources can serve as a alternative when the original training dataset is unavailable. The experiments are conducted on the ResNet18 model for 4-bit weight quantization. The calibration data sources include ImageNet, VOC2012, COCO2017, DeepInv, CIFAR10, and Gaussian noise. The ImageNet is the original training dataset. Both VOC2012 and COCO2017 are public datasets that share certain similarities with ImageNet. The DeepInv is a synthetic dataset generated to imitate the ImageNet dataset by using the DeepInversion method [79]. CIFAR10 is a small-sized public dataset. Gaussian noise consists of random values following Gaussian distributions. For each data source, we randomly select 256 images as the calibration dataset. The accuracy and accuracy degradation (in parentheses) compared with the pre-training accuracy in the respective papers are reported. The results are summarized in Table VI.

In the table, the PPQ method outperforms all other methods across other data sources, with the exception of the Gaussian noise source where the BC method performs the best. On VOC2012 datasets, the PPQ method achieves higher performance. Using COCO2017 and synthetic data result in a slight performance degradation by 0.10% and 0.22%, respectively. Even on the quite different CIFAR10 dataset, the PPQ method only has a 0.46% difference compared to on ImageNet.

Because the calibration dataset is limited, the quantized model may encounter diverse data that are different from calibration dataset. To assess the generalization ability of the proposed method, we evaluate the quantization performance on diverse datasets when using the well-optimized parameters from ImageNet. This experiment is also conducted on ResNet18 for 4-bit weight quantization. To address the conflicting labels for various datasets, we treat the outputs of the unquantized model as labels, which are used to count the accuracy. The experiments are conducted on VOC2012, COCO2017, and DeepInv datasets. We compare the difference between the accuracy on diverse dataset and the accuracy on the ImageNet. The results are summarized in Table VII.

The PPQ method generally outperforms the LAPQ and BC methods, except for on the DeepInv dataset where the LAPQ method performs better. The Bit-Split method exhibits the best performance on DeepInv. However, the PPQ achieves higher performance than the Bit-Split method on the VOC2012 and COCO2017 datasets by 0.29% and 0.50%, respectively. Possibly due to the block-wise optimization strategy, the BRECQ outperforms our method on all datasets. Notably, the PPQ achieves comparable performance to the BRECQ method on VOC2012 and COCO2017 datasets. This experiment demonstrates the good performance of the proposed method when applied to diverse datasets.

VI. Conclusion

In this study, we propose a novel Bayesian-based post-training quantization method. The proposed method estimates the optimal quantization parameters through Maximum A Posterior (MAP) inference by modeling the uncertainty caused by rounding operations with additive uniform noise. The inference model is specifically adapted to be optimized using probabilistic programming techniques with gradient backpropagation. To address the overfitting problem in small datasets, we introduce a Minimum Description Length (MDL) prior that favors low quantization bit widths for quantization step sizes, and we develop a validation procedure to select optimized quantization parameters during the optimization process. The experiments demonstrate that our proposed method performs well when learning from small calibration datasets, achieving nearly state-of-the-art quantization performance even with a calibration dataset of only 64 samples. Furthermore, the proposed method exhibits good generalization ability to optimize with different data sources and strong adaptability to diverse data.

References

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

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[4] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[5] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "Meliusnet: Can binary neural networks achieve net-level accuracy?" *arXiv preprint arXiv:2001.05936*, 2020.

[6] C. Liu, X. Zhang, R. Zhang, L. Li, S. Zhou, D. Huang, Z. Li, Z. Du, S. Liu, and T. Chen, "Rethinking the importance of quantization bias, toward full low-bit training," *IEEE Transactions on Image Processing*, vol. 31, pp. 7006–7019, 2022.

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

[8] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[9] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[10] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[11] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3009–3018.

[12] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7197–7206.

[13] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "Brecq: Pushing the limit of post-training quantization by block reconstruction," *arXiv preprint arXiv:2102.05426*, 2021.

[14] R. Kushol, P. Parnianpour, A. H. Wilman, S. Kalra, and Y.-H. Yang, "Effects of mri scanner manufacturers in classification tasks with deep learning models," *Scientific Reports*, vol. 13, no. 1, p. 16791, 2023.

[15] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[16] G. Baudart, M. Hirzel, and L. Mandel, "Deep probabilistic programming languages: A qualitative study," *arXiv preprint arXiv:1804.06458*, 2018.

[17] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," *Advances in neural information processing systems*, vol. 30, 2017.

[18] J. Achterhold, J. M. Koehler, A. Schmeink, and T. Genewein, "Variational network quantization," in *International Conference on Learning Representations*, 2018.

[19] M. Van Baalen, C. Louizos, M. Nagel, R. A. Amjad, Y. Wang, T. Blankevoort, and M. Welling, "Bayesian bits: Unifying quantization and pruning," *Advances in neural information processing systems*, vol. 33, pp. 5741–5752, 2020.

[20] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[21] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 973–978, 2019.

[22] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382.

[23] J. Zhang, Z. Wang, H. Wang, J. Zhou, and J. Lu, "Anycost network quantization for image super-resolution," *IEEE Transactions on Image Processing*, vol. 33, pp. 2279–2292, 2024.

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[25] W. Xu, F. Li, Y. Jiang, A. Yong, X. He, P. Wang, and J. Cheng, "Improving extreme low-bit quantization with soft threshold," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 4, pp. 1549–1563, 2022.

[26] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1325–1334.

[27] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *International conference on machine learning*. PMLR, 2019, pp. 7543–7552.

[28] G. Shomron, F. Gabbay, S. Kurzum, and U. Weiser, "Post-training sparsity-aware quantization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 737–17 748, 2021.

[29] P. Wang, W. Chen, X. He, Q. Chen, Q. Liu, and J. Cheng, "Optimization-based post-training quantization with bit-split and stitching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 2119–2135, 2023.

[30] T. Chu, Z. Yang, and X. Huang, "Improving the post-training neural network quantization by prepositive feature quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 4, pp. 3056–3060, 2024.

[31] J. Shi, M. Lu, and Z. Ma, "Rate-distortion optimized post-training quantization for learned image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 5, pp. 3082–3095, 2024.

[32] Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11-12, pp. 3245–3262, 2021.

[33] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4466–4475.

[34] X. Wei, R. Gong, Y. Li, X. Liu, and F. Yu, "Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization," *arXiv preprint arXiv:2203.05740*, 2022.

[35] J. Liu, L. Niu, Z. Yuan, D. Yang, X. Wang, and W. Liu, "Pd-quant: Post-training quantization based on prediction difference metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 24 427–24 437.

[36] C. Guo, Y. Qiu, J. Leng, X. Gao, C. Zhang, Y. Liu, F. Yang, Y. Zhu, and M. Guo, "Squant: On-the-fly data-free quantization via diagonal hessian approximation," *arXiv preprint arXiv:2202.07471*, 2022.

[37] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "Zeroq: A novel zero shot quantization framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 169–13 178.

[38] K. Choi, D. Hong, N. Park, Y. Kim, and J. Lee, "Qimera: Data-free quantization with synthetic boundary supporting samples," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 835–14 847, 2021.

[39] S. Xu, S. Zhang, J. Liu, B. Zhuang, Y. Wang, and M. Tan, "Generative data free model quantization with knowledge matching for classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 12, pp. 7296–7309, 2023.

[40] H. Luo, S. Zhang, Z. Zhuang, J. Mai, M. Tan, and J. Zhang, "Learning to generate diverse data from a temporal perspective for data-free quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 10, pp. 9484–9498, 2024.

[41] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[42] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[43] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *IEEE Transactions on instrumentation and measurement*, vol. 45, no. 2, pp. 353–361, 1996.

[44] E. Agustsson and L. Theis, "Universally quantized neural compression," *Advances in neural information processing systems*, vol. 33, pp. 12 367–12 376, 2020.

[45] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[46] L. Zhou, C. Cai, Y. Gao, S. Su, and J. Wu, "Variational autoencoder for low bit-rate image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2617–2620.

[47] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2743–2760, 1998.
[48] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, 2015.
[49] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," *arXiv preprint arXiv:1711.10604*, 2017.
[50] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[51] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.
[52] M. Figurnov, S. Mohamed, and A. Mnih, "Implicit reparameterization gradients," *Advances in neural information processing systems*, vol. 31, 2018.
[53] D. Piponi, D. Moore, and J. V. Dillon, "Joint distributions for tensorflow probability," *arXiv preprint arXiv:2001.11819*, 2020.
[54] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[55] A. Gunjal and G. Durrett, "Drafting event schemas using language models," *arXiv preprint arXiv:2305.14847*, 2023.
[56] D. Peek, M. P. Skerritt, and S. Chalup, "Synthetic data generation and deep learning for the topological analysis of 3d data," in *2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2023, pp. 121–128.
[57] D. Duvenaud, D. Maclaurin, and R. Adams, "Early stopping as nonparametric variational inference," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 1070–1077.
[58] A. Makarova, H. Shen, V. Perrone, A. Klein, J. B. Faddoul, A. Krause, M. Seeger, and C. Archambeau, "Automatic termination for hyperparameter optimization," in *International Conference on Automated Machine Learning*. PMLR, 2022, pp. 7–1.
[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.
[60] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[63] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
[64] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
[65] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[66] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.
[67] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.
[68] Y. Shang, G. Liu, R. R. Kompella, and Y. Yan, "Enhancing post-training quantization calibration through contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 921–15 930.
[69] Y. Jeon, C. Lee, E. Cho, and Y. Ro, "Mr. biq: Post-training non-uniform quantization based on minimizing the reconstruction error," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 329–12 338.
[70] X. Ding, X. Liu, Z. Tu, Y. Zhang, W. Li, J. Hu, H. Chen, Y. Tang, Z. Xiong, B. Yin *et al.*, "Cbq: Cross-block quantization for large language models," *arXiv preprint arXiv:2312.07950*, 2023.
[71] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.
[72] X. Wei, Y. Zhang, X. Zhang, R. Gong, S. Zhang, Q. Zhang, F. Yu, and X. Liu, "Outlier suppression: Pushing the limit of low-bit transformer language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 402–17 414, 2022.
[73] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu, "Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling," *arXiv preprint arXiv:2304.09145*, 2023.
[74] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization," in *European conference on computer vision*. Springer, 2022, pp. 191–207.
[75] Y. Ding, H. Qin, Q. Yan, Z. Chai, J. Liu, X. Wei, and X. Liu, "Towards accurate post-training quantization for vision transformer," in *Proceedings of the 30th ACM international conference on multimedia*, 2022, pp. 5380–5388.
[76] Z. Li, J. Xiao, L. Yang, and Q. Gu, "Repq-vit: Scale reparameterization for post-training quantization of vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 227–17 236.
[77] N. Ranjan and A. Savakis, "Lrp-qvit: Mixed-precision vision transformer quantization via layer-wise relevance propagation," *arXiv preprint arXiv:2401.11243*, 2024.
[78] Y. Zhong, Y. Huang, J. Hu, Y. Zhang, and R. Ji, "Towards accurate post-training quantization of vision transformers via error reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.
[79] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8715–8724.

Kui Liu received the B.S. and M.S. degrees from Chongqing University, Chongqing, China, in 2018 and 2021, respectively. He is currently pursuing Ph.D. degree with IMEC & Ghent University, Ghent, Belgium. His research interests include model compression, Bayesian learning, probabilistic programming.

Bart Goossens received the master's degree in computer science engineering from Ghent University, Belgium, in 2006, and the PhD degree from Ghent University, Belgium, in 2010. He is an associate professor in the Image Processing and Interpretation Group of imec-Ghent University, where he currently supervises research on image/video processing, computer vision, AI and heterogeneous platforms mapping tools. He received the IEEE Best Paper Award at ISPA 2024, the IEEE SPS 2023 Outstanding Editorial Board Member Award (for editorial board service at the IEEE Transactions on Image Processing), the annual Scientific Prize IBM Belgium for Informatics in 2011 and the Barco/FWO prize in 2006. He currently serves as an associate editor for the IEEE Transactions on Image Processing.

Tom De Schepper obtained in 2019 the degree of PhD in Computer Science from the University of Antwerp, Belgium. His PhD was centered around network management, distributed systems, and artificial intelligence. Afterwards he was part of the research group IDLab, as a senior research and program manager, leading a team of multiple junior and senior researchers. He was involved in various national and international collaborations with other research institutes and companies. For instance, he was appointed challenge manager of the "Human-like AI" grand challenge in the Flanders

AI research program, coordinating research across multiple Flemish research teams of different universities. In 2022, he joined the research center imec as a Principal Researcher, acting as the Scientific lead for AI-based sensing (incl. sensor fusion and edge AI) within imec's AI & Algorithms department. He is a co-author of more than 50 internationally peer reviewed publications.

Wilfried Philips was born in Aalst, Belgium, in 1966. He received the Diploma in electrical engineering and the Ph.D. degree in applied sciences from Ghent University, Ghent, Belgium, in 1989 and 1993, respectively. He is currently a Senior Full Professor with the Department of Telecommunications and Information Processing, Ghent University, where he heads the Image Processing and Interpretation Research Group. He also leads the activities in image processing and sensor fusion within the research institute IMEC. His main research interests include image and video quality improvement and estimation, real-time computer vision, and sensor data processing. He is also a Cofounder of the Senso2Me Company, which provides Internet of Things Solutions for elderly care.