MetaAug: Meta-Data Augmentation for Post-Training Quantization

Cuong Pham[1], Hoang Anh Dung[1], Cuong C. Nguyen[2], Trung Le[1], Dinh Phung[1,3], Gustavo Carneiro[2], and Thanh-Toan Do[1]

[1] Department of Data Science and AI, Monash University, Australia
[2] Centre for Vision, Speech and Signal Processing, University of Surrey, UK
[3] VinAI, Vietnam
{cuong.pham1, hoang.dung, trunglm, dinh.phung, toan.do}@monash.edu
{c.nguyen, g.carneiro}@surrey.ac.uk

Abstract. Post-Training Quantization (PTQ) has received significant attention because it requires only a small set of calibration data to quantize a full-precision model, which is more practical in real-world applications in which full access to a large training set is not available. However, it often leads to overfitting on the small calibration dataset. Several methods have been proposed to address this issue, yet they still rely on only the calibration set for the quantization and they do not validate the quantized model due to the lack of a validation set. In this work, we propose a novel meta-learning based approach to enhance the performance of post-training quantization. Specifically, to mitigate the overfitting problem, instead of only training the quantized model using the original calibration set without any validation during the learning process as in previous PTQ works, in our approach, we both train and validate the quantized model using two different sets of images. In particular, we propose a meta-learning based approach to jointly optimize a transformation network and a quantized model through bi-level optimization. The transformation network modifies the original calibration data and the modified data will be used as the training set to learn the quantized model with the objective that the quantized model achieves a good performance on the original calibration data. Extensive experiments on the widely used ImageNet dataset with different neural network architectures demonstrate that our approach outperforms the state-of-the-art PTQ methods.

Keywords: Network Quantization · Post Training Quantization · Meta Learning · Deep Neural Networks

1 Introduction

Deep neural networks (DNNs) have received a substantial amount of attention due to their state-of-the-art performance in various tasks. However, deploying these networks on resource-constrained devices is challenging due to the limited computational resources and memory footprint. To make DNNs more efficient,

network quantization [3, 5, 12, 26, 27, 45] has been extensively studied due to its computational and storage benefits. Quantization is the process of reducing the precision of the weights and activations of DNNs. Depending on the available training data, network quantization can be divided into two main categories: quantization-aware training (QAT) [5, 6, 8, 11, 31, 38, 46] and post-training quantization (PTQ) [15, 20, 23, 26, 44]. Although QAT generally results in better performance compared to PTQ and can reduce the gap to full-precision accuracy for low-bit quantization, it requires a large training set to retrain DNNs on the targeting dataset. This may not be practical for many real-world applications where a large training dataset is unavailable or access to it is restricted due to security and privacy concerns.

To tackle this problem, PTQ has been investigated because it only employs a small calibration dataset to quantize a well-trained full-precision model. However, this approach often results in overfitting to the used small calibration set [23, 44, 50]. Various methods have been proposed to mitigate this overfitting issue. In QDrop [44], the authors propose to mitigate overfitting in PTQ by randomly dropping quantized activations. In [50], the authors utilize activation regularization by minimizing the difference between the intermediate features of the full-precision model and the quantized model. In PD-Quant [23], the authors indicate the performance degradation in PTQ due to a severe overfitting on the calibration set and they also adopt activation regularization to counteract overfitting. In addition, they introduce activation distribution correction as regularization to further alleviate overfitting by encouraging the distributions of quantized activations to match the batch normalization (BN) statistics from the BN layers of the full-precision model. Although different strategies have been proposed, they all rely on the original calibration data for training the quantized model and they do not have a validation set to validate the quantized model during the quantization process. This could lead the quantized model to be prone to overfitting on the calibration set.

Different from previous works [23, 44, 50] in PTQ that use the calibration set for training and do not have a validation set to validate the quantized model, in this work, we propose to perform the quantization using two different sets – a modified version of the calibration set is used as the training data for learning the quantized model, while the original calibration data is used as the validation set to validate the quantized model. The modified data is produced by a learnable transformation network that takes the original calibration data as input. Our work aims to jointly optimize both the transformation network and the quantized network with the objective that they lead to a good performance of the quantized network on the validation set, i.e., the original calibration set. However, this aim is nontrivial. This is because the problem is a nested optimization in which the optimization for the transformation network is to minimize a validation loss of the quantized network while the quantized network itself is subjected to another optimization with some quantization loss.

To tackle this challenge, we propose a novel meta-learning based PTQ approach in which the transformation network and the quantized network are

jointly optimized through a bi-level optimization. A noticeable challenge in this approach is the possibility of the transformation network to be degenerated into an identity mapping. Consequently, such scenario can result in overfitting in the quantization process, as the training and the validation of the quantized model use the same original calibration data. To prevent this situation, we deeply investigate approaches to make the transformation network capable of preserving the information of the original calibration data while still giving it the flexibility to avoid being a trivial (i.e., an identity) transformation. Specifically, we investigate three different losses for semantic preservation, including a probabilistic knowledge transfer loss. This encourages the transformation network to capture the feature distributions of the original calibration data which consequently preserves the information of the calibration data. In addition, we also propose using a margin loss to discourage the transformation network from being a trivial transformation. We validate our proposed approach on the widely used ImageNet dataset with different neural network architectures by comparing it with state-of-the-art methods. The extensive empirical results demonstrate that our method outperforms the state-of-the-art PTQ methods.

Our contributions can be summarized as follows: ❶ We propose a novel meta-learning based method to mitigate the overfitting problem in PTQ. The proposed approach jointly optimizes a transformation network and a quantized model. During the learning process, the outputs of the transformation network and the original data are used for training and validating the quantized model, respectively. To the best of our knowledge, this is the first work that tackles the overfitting problem in PTQ through a meta-learning bi-level optimization approach.❷ We investigate different losses for training the transformation network such that the outputs of the network preserve the feature information of the original calibration data. Furthermore, we also propose using a margin loss to discourage the transformation network from being an identity mapping. ❸ We validate our proposed approach on the widely used ImageNet dataset across various neural network architectures. Extensive experiments demonstrate that the proposed method outperforms the state-of-the-art PTQ methods.

2 Related work

Uniform quantization. To uniformly quantize a tensor w to b bit-width, the support space is uniformly discretized into $2^b - 1$ even intervals. As a result, the original 32-bit single-precision value is mapped to an unsigned integer within the range of $[0,\ 2^b - 1]$, or a signed integer within the range of $[-2^{b-1},\ 2^{b-1} - 1]$. Supposed that Q_b is the quantization function with a bit-width of b, the quantization function Q_b is defined as follows:

$$\hat{w} = Q_b(w; s) = s \times \text{clip}\left(\left\lfloor \frac{w}{s} \right\rceil, n, p\right), \tag{1}$$

where s represents the scaling factor, $\lfloor . \rceil$ denotes the rounding-to-nearest function, and clip() represents the clipping function. For unsigned data (e.g., activations with ReLU or Sigmoid) $n = 0$, $p = 2^b - 1$, and for signed data (e.g.,

weights) $n = -2^{b-1}$, $p = 2^{b-1} - 1$. In PTQ, rounding-to-nearest is the most common rounding function by minimizing the quantization error. However, the most recent state-of-the-art approaches [15,20,23,26,44] have shown that a learnable rounding function can improve the performance of quantized models. The quantization function Q_b in those studies is defined as:

$$\hat{w} = Q_b(w; s, v) = s \times \text{clip}\left(\left\lfloor \frac{w}{s} \right\rfloor + h(v), n, p\right) \quad \text{s.t.: } v \in \{0, 1\}, \tag{2}$$

where $h(v)$ is a learnable function that maps the value of v to either 0 or 1. Note that during training, the scaling factor s is fixed in AdaRound [26], while being learned simultaneously with the rounding function $h(v)$ in Genie [15]. In our work, we adopt the Genie [15] approach for weight quantization and LSQ [8] for activation quantization.

Post training quantization (PTQ). This quantization approach has gained considerable attention recently because it does not require access to large amounts of data and can operate effectively with minimal or even unlabeled training data. This method is particularly useful when full access to training data is not possible. In addition, it is useful for large models that are not suitable for QAT due to their substantial training time. In AdaRound [26], the authors propose using a learnable rounding function instead of the traditional rounding-to-nearest approach to quantize the model layer by layer. Based on this, BRECQ [20] further improves the performance of PTQ by proposing block reconstruction (e.g., 4 blocks in ResNet18 [13]) that considers the dependency of layers' outputs in each block of the neural network. In [24], the authors address the problem of oscillation in PTQ. They propose a method to identify blocks within a network that should be jointly optimized and quantized. In QDrop [44], their framework exploits a mechanism randomly dropping quantized activations to improve the flatness of the quantized model. In [50], an activation regularization is proposed, by minimizing the difference between the intermediate features after the activation function of the quantized model and the full-precision model. In addition to activation regularization, another method named PD-Quant [23] demonstrates performance improvement by correcting the feature distribution of calibration data to follow the feature distribution of full-training data based on batch normalization (BN) statistics from the BN layers of the full-precision model. In Genie [15], the authors propose to learn the scale and rounding functions simultaneously to further improve the performance of PTQ. Another approach for PTQ is Bit-Shrinking [22], which incorporates sharpness-aware minimization into the quantization process. In that method, the authors suggest progressively reducing the bit-width of quantized models to limit the instantaneous sharpness of the objective function. It is worth noting that all the mentioned methods only rely on the original calibration data for training the quantized model. They do not have a validation set to validate the quantized model during the quantization process. This could lead the quantized model to be prone to overfitting on the calibration set.

Meta-learning. Meta-learning methods can be divided into three categories: optimization-based, model-based, and metric-based methods. Optimization-based meta-learning investigates the optimization in the task adaptation step and uses training tasks to improve that optimization (e.g., learn a good learning rate [21], model initialization [10], updating rule [32] or even a data-driven optimizer [2]). Among many optimization-based meta-learning methods, MAML [10] is one of the most popular ones. MAML aims to learn a meta-model that can quickly adapt to new tasks with few training examples. Since then, many variants of this optimization-based approach have been proposed to further enhance the performance [1, 9, 16, 29]. On the other hand, model-based meta-learning models, such as Memory-Augmented Neural Networks (MANNs) [36] and Recurrent Meta-Learners framework [7, 42], maintain an internal representation of a task during training. This internal state is periodically updated based on new inputs and makes great contribution to the model output. Finally, the last branch of meta-learning methods – metric-based frameworks [19, 37, 39–41], are designed to learn an embedding function to map all data points to a metric embedding space. Overall, despite demonstrating the ability to generalize the model over unseen data, there is still not enough attention regarding the applicability in PTQ of meta-learning.

Meta-Learning for Network Quantization. Several works [4,17,43,47] have utilized meta-learning for quantization. In MetaQuantNet [43], the authors propose a framework that can automatically search for the best quantization policy with meta-learning before using that policy for quantization. On the other hand, MEBQAT [47] attempts to leverage the meta-learning mechanism to optimize a mixed-precision quantization model capable of adapting to different bit-width scenarios quickly without hurting the model's performance. Another work named MetaMix [17] points out the activation instability problem in existing methods for mixed-precision quantization and aims to tackle this problem with meta-learning. However, these methods focus on quantization-aware training, while our work focuses on mitigating overfitting in post-training quantization. Additionally, all of these methods only leverage meta-learning to improve their quantization mechanisms without considering the impact of calibration data to their frameworks. To the best of our knowledge, our work is the first to leverage meta-learning in the context of post-training quantization, from the perspective of data optimization.

3 Proposed method

3.1 Meta-learning formulation for PTQ

Let $S = \{x_i\}_{i=1}^{N}$ be the calibration set. Given a sample $x_i \sim S$, consider a full-precision model θ_{FP} and a quantized model θ_Q, our objective is to learn a transformation network T that modifies the calibration sample x_i into an adaptive sample $T(x_i)$ beneficial for model generalization. The data sample $T(x_i)$ outputted by T is then utilized to optimize the quantized network θ_Q to get

a model $\widehat{\theta}_Q$ after a number of gradient descent steps. The optimal T is then determined based on performance of the model $\widehat{\theta}_Q$ on the original data set $S^v = \{x_i^v\}_{i=1}^N$ (in this context, $S^v = S$). The bi-level objective function is defined as:

$$T^* = \arg\min_T \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v) \tag{3}$$

$$\text{s.t.: } \widehat{\theta}_{\text{Q}} = \arg\min_{\theta_Q} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_Q(\theta_{\text{Q}}, T(x_i)). \tag{4}$$

The objective function in Eq. (3) represents a bi-level optimization problem, typically solved in two stages. The first stage presented in Eq. (4) involves optimizing θ_Q using the modified data $\{T(x_i)|i = 1, 2, .., N\}$. This stage can be addressed using gradient-based optimization methods, such as SGD or Adam, as follows:

$$\widehat{\theta}_{\text{Q}} = \theta_{\text{Q}} - \frac{\eta}{N} \sum_{i=1}^{N} \boldsymbol{\nabla}_{\theta_{\text{Q}}} \mathcal{L}_Q(\theta_{\text{Q}}, T(x_i)), \tag{5}$$

where η is the learning rate to update $\widehat{\theta}_{\text{Q}}$.

The second stage involves updating T based on the model $\widehat{\theta}_{\text{Q}}$, focusing on the performance evaluated on the original data x_i^v. This update corresponds to the upper-level optimization. This can be expressed as follows:

$$T \leftarrow T - \frac{\gamma}{N} \sum_{i=1}^{N} \boldsymbol{\nabla}_T \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v) \tag{6}$$

where γ is the learning rate to update T.

As shown in Eq. (6), optimizing T requires calculating the gradient of the validation loss $\mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v)$ with respect to T. Using the chain rule, the computation can be performed as follows:

$$\begin{aligned}
\boldsymbol{\nabla}_T \mathcal{L}_{\text{val}} \left(\widehat{\theta}_{\text{Q}}, x_i^v\right) &= \boldsymbol{\nabla}_T^\top \widehat{\theta}_{\text{Q}} \times \boldsymbol{\nabla}_{\widehat{\theta}_{\text{Q}}} \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v) \\
&= \boldsymbol{\nabla}_T^\top \left[\theta_{\text{Q}} - \frac{\eta}{N} \sum_{j=1}^{N} \boldsymbol{\nabla}_{\theta_{\text{Q}}} \mathcal{L}_Q(\theta_{\text{Q}}, T(x_j))\right] \times \boldsymbol{\nabla}_{\widehat{\theta}_{\text{Q}}} \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v) \\
&= -\frac{\eta}{N} \boldsymbol{\nabla}_T (\sum_{j=1}^{N} \boldsymbol{\nabla}_{\theta_{\text{Q}}} \mathcal{L}_{\text{Q}}(\theta_{\text{Q}}, T(x_j)))^\top \times \boldsymbol{\nabla}_{\widehat{\theta}_{\text{Q}}} \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, x_i^v),
\end{aligned} \tag{7}$$

where $^\top$ denotes the transpose operator.

Regarding $\mathcal{L}_Q$ in (4). We adopt the block-wise [20] quantization method to sequentially quantize the full-precision model θ_{FP} to get the quantized model

θ_{Q}. Given the pre-trained full-precision model θ_{FP} consisting of L blocks, we sequentially quantize the model block by block and update the transformation network T to minimize the validation loss of $\widehat{\theta}_{\mathrm{Q}}$ on the original calibration data S. The loss in Eq. (4) updating the l^{th} block of the model θ_{Q} to obtain the model $\widehat{\theta}_{\mathrm{Q}}$ is defined as follows:

$$\mathcal{L}_Q(\theta_{\mathrm{Q}}, T(S)) = \frac{1}{N}\sum_{i=1}^{N}\left\|A^l_{FP}(T(x_i)) - A^l_Q(T(x_i))\right\|^2, \tag{8}$$

where $A^l(x_i)$ and and $A^l_Q(x_i)$ are the activations of the l^{th} block of the full-precision model θ_{FP} and the quantized model θ_{Q} for sample $T(x_i)$, respectively.

Regarding $\mathcal{L}_{\mathrm{val}}$ ***in*** *(3).* As shown in (3), our goal is to minimize the validation loss of $\widehat{\theta}_{\mathrm{Q}}$ on the original data. Therefore, at the validation step, we validate the quantized model $\widehat{\theta}_{\mathrm{Q}}$ on the original calibration set $S = \{x_i\}_{i=1}^N$. We adopt Kullback-Leibler divergence loss to validate the quantized model $\widehat{\theta}_{\mathrm{Q}}$ on the original calibration data S, which is defined as follows:

$$\mathcal{L}_{\mathrm{val}}(\widehat{\theta}_{\mathrm{Q}}, S) = \frac{1}{N}\sum_{i=1}^{N}\mathrm{KL}\left[\sigma(f_{\theta_{\mathrm{FP}}}(x_i))\|\sigma(f_{\widehat{\theta}_{\mathrm{Q}}}(x_i))\right], \tag{9}$$

where f is output of the model of interest and $\sigma(.)$ denotes the softmax operator.

3.2 Transformation T and regularizations to the modified images

In this section, we discuss the definition of transformation network T and objective functions to update T. The transformation network could be parameterized by an autoencoder, a UNet, or any other transformation network. In this work, we use the UNet [33] as the transformation network. The UNet is a widely used architecture for image-to-image translation tasks, consisting of an encoder and a decoder. The encoder is used to extract features from the input image, and the decoder is used to generate the output image. The UNet model has advantages over other autoencoders in retaining the fine feature information of the input image because it includes residual connections between the encoder and decoder. On the one hand, we expect the generated images $T(x_i)$ to retain the information of original images x_i. On another hand, the transformation network T should not be degenerated into a trivial solution i.e., an identity mapping, as it would have no effect on overfitting reduction. We investigate different objective functions to update T.

Information Preservation. Given the original calibration set S, we have a corresponding transformed image set $S^{(g)} = \{T(x_i)|x_i \sim S, 1 \leq i \leq N\}$. To transfer the information of images from S to the generated set $S^{(g)}$, we investigate different losses for this purpose including a Mean Square Error loss (MSE),

a Kullback–Leibler (KL) divergence loss, and a distribution preservation loss. The MSE between outputs from the full-precision model of original images and generated images is defined:

$$\mathcal{L}_{\text{MSE}}(T,S) = \frac{1}{N}\sum_{i=1}^{N} \|f_{\theta_{\text{FP}}}(x_i) - f_{\theta_{\text{FP}}}(T(x_i))\|^2. \tag{10}$$

The KL loss between outputs from the full precision model of original images and generated images is defined:

$$\mathcal{L}_{\text{KL}}(T,S) = \frac{1}{N}\sum_{i=1}^{N} \text{KL}\left[\sigma(f_{\theta_{\text{FP}}}(x_i)) \| \sigma(f_{\theta_{\text{FP}}}(T(x_i))\right] \tag{11}$$

It is worth noting that the losses (10) and (11) only consider pairwise distances between corresponding features from the full-precision (FP) and quantized models, without considering the information between samples. Therefore, we also consider another information preservation loss that aims to retain the whole dataset's distribution information by leveraging the distribution probabilistic loss that has been used in [30]. Specifically, we first estimate the conditional probability density of any two data points within the feature space [25,30], which is formulated as:

$$\mathcal{P}_{i|j} = \frac{K(f_{\theta_{\text{FP}}}(x_i), f_{\theta_{\text{FP}}}(x_j))}{\sum_{\substack{k=1 \\ k\neq j}}^{N} K(f_{\theta_{\text{FP}}}(x_k), f_{\theta_{\text{FP}}}(x_j))}, \tag{12}$$

where $K(a,b)$ is a kernel function and $\mathcal{P}_{i|j}$ is the probability of x_i given x_j. Following PKT [30], we adopt the cosine similarity metric $K(a,b) = \frac{1}{2}(\frac{a^T b}{\|a\|\|b\|}+1)$ as kernel function. To encourage the feature distribution matching between the original dataset S and the generated dataset $S^{(g)}$, original image x_i should share the same probability distribution with its corresponding generated image $T(x_i)$, so the distribution preservation loss $\mathcal{L}_{DP}$ is defined as:

$$\mathcal{L}_{DP}(T,S) = \frac{1}{N}\sum_{i=1}^{N} \text{KL}\left[\mathcal{P}_i \| \mathcal{P}_i^{(g)}\right], \tag{13}$$

where $\mathcal{P}_i$ and $\mathcal{P}_i^{(g)}$ are the conditional probability distributions of the extracted features from the full precision model of original image x_i and generated image $T(x_i)$, respectively.

Identity Prevention. To encourage the transformation network not to be an identity, we propose using the following loss:

$$\mathcal{L}_{\text{margin}}(T,S) = \frac{1}{N}\sum_{i=1}^{N} \max\left(0, \epsilon - \frac{1}{M}\|x_i - T(x_i)\|^2\right), \tag{14}$$

where ϵ is a threshold to encourage that the difference between the generated data and the original data is not lower than the threshold, and M is the total number of pixels of image x_i.

Algorithm 1 Data modification for post-training quantization.

procedure TRAIN(θ_{FP},S)
  ▷ *θ_{FP}: weight of the full-precision model.* ◁
  ▷ *L: Number of blocks in the full-precision model.* ◁
  ▷ *S: Calibration data.* ◁
  ▷ *N_T: Number of iterations to update T.* ◁
  ▷ *N_Q: Number of iterations to quantize model.* ◁
  ▷ *T: Transformation network to modify calibration dataset S.* ◁
  Initialize the quantized model θ_Q from θ_{FP} using LAPQ [28].
  Warm up the transformation network T.
  for $l = 1$ to L **do**
    for $t = 1$ to N_T **do**
      Sample a mini-batch: $\mathbb{B} = \{x_i : x_i \sim \mathcal{S}\}$
      Modify $\mathbb{B}$ with the transformation network T to get $T(\mathbb{B}) = \{T(x_i)\}_{i=1}^{|\mathbb{B}|}$
      ▷ *Forward pass and update the quantized model using modified data.* ◁
      Compute: $\mathcal{L}_Q(\theta_{\text{Q}}, T(\mathbb{B})) = \frac{1}{|\mathbb{B}|}\sum_{i=1}^{|\mathbb{B}|} ||A_{FP}^l(T(x_i)) - A_Q^l(T(x_i))||^2$ ▷ *Eq.* (8)
      Update $\widehat{\theta}_Q$: $\widehat{\theta}_Q \leftarrow \text{Adam}(\mathcal{L}_Q(\theta_{\text{Q}}, T(\mathbb{B})))$
      ▷ *Validate $\widehat{\theta}_Q$ on the original calibration data.* ◁
      Sample a mini-batch data: $\mathbb{B}^v = \{x_i^v : x_i^v \sim \mathcal{S}\}$
      Compute: $\mathcal{L}_T(T, \mathbb{B}^v)$ ▷ *Eq.* (15)
      Update T: $T \leftarrow \text{Adam}(\mathcal{L}_T(T, \mathbb{B}^v))$
    ▷ *Quantize l^{th} block of θ_Q using the original calibration data $\mathcal{S}$ and modified data with the learned T.* ◁
    for $t = 1$ to N_Q **do**
      Sample a mini-batch: $\mathbb{B}_q = \{x_{qi} : x_{qi} \sim \mathcal{S}_q = T(\mathcal{S}) \cup \mathcal{S}\}$
      Compute: $\mathcal{L}_Q(\theta_{\text{Q}}, \mathbb{B}_q) = \frac{1}{|\mathbb{B}_q|}\sum_{i=1}^{|\mathbb{B}_q|} ||A_{FP}^l(x_{qi})) - A_Q^l(x_{qi})||^2$
      Update: $\theta_Q \leftarrow \text{Adam}(\mathcal{L}_Q(\theta_{\text{Q}}, \mathbb{B}_q))$
  return quantized model θ_{Q} and T.

Overall loss for training T. Combine objective loss in Eq. (9), and in Eq. (14) with either objective losses in Eq. (10), Eq. (11), Eq. (13), we have the final combination loss to update T as follows:

$$\mathcal{L}_T(T, S) = \lambda_1 \mathcal{L}_{\text{val}}(\widehat{\theta}_{\text{Q}}, S) + \lambda_2 \mathcal{L}_{\text{margin}}(T, S) + \lambda_3 \mathcal{L}_*(T, S), \tag{15}$$

where $\mathcal{L}_* \in \{\mathcal{L}_{\text{MSE}}, \mathcal{L}_{\text{KL}}, \mathcal{L}_{\text{DP}}\}$ and λ_1, λ_2 and λ_3 are hyper-parameters.

The overall algorithm of our proposed method is presented in Algorithm 1.

Table 1: Top-1 classification accuracy (%) with the ResNet-18 architecture with different combinations of proposed losses evaluated on ImageNet dataset.

Setting	$\mathcal{L}_{\text{val}}$	$\mathcal{L}_{\text{MSE}}$	$\mathcal{L}_{\text{KL}}$	$\mathcal{L}_{\text{DP}}$	$\mathcal{L}_{\text{margin}}$	ResNet-18 W2A2
Genie-M [15]						53.71
(a)	✓					53.45
(b)	✓	✓				53.64
(c)	✓		✓			53.89
(d)	✓			✓		54.09
(e)	✓			✓	✓	**54.22**

4 Experiments

4.1 Experimental setup

Datasets and network architectures. We validate the proposed method on the ImageNet dataset [34]. Following previous PTQ works [15, 20, 22, 23, 26, 44], the calibration set used for training quantized models contains 1,024 images from the training set of the ImageNet dataset. The validation set of the ImageNet dataset containing 50,000 images is used as the test set. Following previous PTQ works [15,20,22,23,26,44], we evaluate our approach on the widely used network architectures including ResNet-18 [13], ResNet-50 [13], and MobileNetV2 [35].

Implementation details. We utilize the UNet [33] as a transformation network to modify the calibration dataset. We use the Adam optimizer [18] with a learning rate of 5×10^{-6} to update the transformation network. This network is trained for 500 iterations with a batch size of 32. For the quantization of weights and activations, we follow current state-of-the-art approaches PTQ [15, 20, 23, 26, 44]. Specifically, for weight quantization, we learn both the scaling factor and rounding function following the Genie approach [15]. For activation quantization, we adopt the LSQ [8]. We also keep the first and last layers at 8 bits as it does not increase much memory storage and helps prevent significant performance degradation [31]. The quantized model θ_Q is initialized from the full-precision model using LAPQ [28] as previous works [15, 20, 23, 26, 44]. We use 2×10^4 iterations to quantize each block of the quantized model. When updating the transformation network T, to compute $\boldsymbol{\nabla}_T \mathcal{L}_{\text{val}}$ in Eq. (7), we utilize the *higher*[4] library. We set the margin parameter ϵ in Eq. (14) to 0.3 for experiments with ResNet-50, and 0.1 for experiments with ResNet-18 and MobileNetV2. We set the hyper-parameters $\lambda_1 = 5$, and $\lambda_2 = 0.5$. We set λ_3 to 1, 5, and 3×10^4 for the $\mathcal{L}_{MSE}$ in Eq. (10), $\mathcal{L}_{KL}$ in Eq. (11), and $\mathcal{L}_{DP}$ in Eq. (13), respectively.

4.2 Ablation studies

Comparitions of information preservation losses. We conduct ablation studies to compare the three different information preservation losses Eq. (10),

[4] https://github.com/facebookresearch/higher

Eq. (11), and Eq. (13). As shown in (15), the final loss for updating the transformation network T is a combination of three different losses, consisting of the validation loss $\mathcal{L}_{val}$, identity prevention loss $\mathcal{L}_{margin}$, and one of the three information preservation losses. We conduct experiments on the ResNet-18 model for the 2/2 bit-width setting. The results are presented in Table 1. The results show that the classification accuracy decreases compared to the Genie baseline [15] when using only $\mathcal{L}_{val}$ (setting (a)). Meanwhile, combining $\mathcal{L}_{val}$ with $\mathcal{L}_{DP}$ (setting (d)) results in improvements of 0.45% and 0.2% compared to the combinations of $\mathcal{L}_{val}$ with $\mathcal{L}_{MSE}$ (setting (b)), and $\mathcal{L}_{val}$ with $\mathcal{L}_{KL}$ (setting (c)), respectively. Furthermore, using additional $\mathcal{L}_{margin}$ (settings (e)) results in even further improvements. This shows that both $\mathcal{L}_{margin}$ and $\mathcal{L}_{DP}$ are essential for the final results. For the remaining results in the following sections, $\mathcal{L}_{DP}$ is used in the $\mathcal{L}_T$ (Eq. (15)). The ablation studies of the hyper-parameters are provided in the supplementary material.

4.3 Comparisons with the state of the art

In this section, we compare our proposed method against the state-of-the-art methods for PTQ, including AdaRound [26], BRECQ [20], QDrop [44], PD-Quant [23], Genie-M [15], and Bit-Shrinking [22]. The results of competitors are cited from the corresponding papers except for the 2/2 setting of Genie-M [15] with the MobileNetV2 network which is reproduced by their official release code. Table 2 presents the comparative results of our proposed MetaAug and other state-of-the-art approaches when evaluating on the ImageNet dataset. It is clear that our proposed method outperforms the other methods across various network architectures. Compared to the current state-of-the-art approaches, Genie-M [15], our proposed method consistently outperforms Genie-M in all bit-width settings. The improvement is clearer in the 2/2 settings, with an improvement of 0.51%, 0.59%, and 0.72% for ResNet-18, ResNet-50, and MobileNetV2, respectively. When activation of the second layer is kept at 8-bit precision, following BRECQ [20] setting, our proposed method outperforms the current state-of-the-art, Bit-Shrinking [22] in all bit-width settings except for the ResNet-50 in 4/4 setting. The improvement is clearer with the highest improvement over Bit-shrinking being 1.47% for ResNet-50 in the 2/2 setting, which confirms the effectiveness of our proposed approach.

Mitigating overfitting. To investigate the benefits of our approach in addressing the overfitting problem, we conduct experiments demonstrating the performance of our methods over the calibration set (i.e., the train set) and the test set, compared to other state-of-the-art methods, including PD-Quant [23], QDrop [44], and Genie [15]. The results are presented in Table 3. It is clear that our proposed method not only achieves the highest accuracy on the test set compared to other models but also yields the smallest train-test accuracy gap. Compared to QDrop [44], while there is a marginal difference between our proposed and QDrop [44] in terms of the train-test accuracy gap (16.98% versus

Table 2: Comparisons of Top-1 classification accuracy (%) with the state of the art on ImageNet dataset. The notation $*$ indicates that the input (activation) of the second layer is maintained at 8-bit precision following BRECQ [20] setting. The result denoted with ‡ is reproduced using the official released code of the corresponding paper.

Method	**Bit-width** (W/A)	**ResNet-18**	**ResNet-50**	**MobileNetV2**
FP	32/32	71.01	76.63	72.20
AdaRound [26]	4/4	67.96	73.88	61.52
BRECQ* [20]		69.60	75.05	66.57
QDrop [44]		69.10	75.03	67.89
QDrop* [44]		69.62	75.45	68.84
PD-Quant [23]		69.23	75.16	68.19
Genie-M [15]		69.35	75.21	68.65
Bit-Shrinking* [22]		69.94	**76.04**	69.02
MetaAug (Ours)		69.48	75.29	68.76
MetaAug* (Ours)		**69.97**	75.78	**69.22**
AdaRound [26]	3/3	64.14	68.40	41.52
BRECQ* [20]		64.80	70.29	53.34
QDrop [44]		65.56	71.07	54.27
QDrop* [44]		66.75	72.38	57.98
Genie [15]		66.16	71.61	57.54
Bit-Shrinking* [22]		67.12	72.91	58.66
MetaAug (Ours)		66.37	71.73	57.77
MetaAug* (Ours)		**67.66**	**73.04**	**59.87**
BRECQ* [20]	2/4	64.80	70.29	53.34
QDrop [44]		64.66	70.08	52.92
QDrop* [44]		65.25	70.65	54.22
PD-Quant [23]		65.17	70.77	55.17
Genie-M [15]		65.77	70.51	56.38
Bit-Shrinking* [22]		65.77	71.11	54.88
MetaAug (Ours)		66.01	70.76	56.45
MetaAug* (Ours)		**66.48**	**71.48**	**56.65**
BRECQ* [20]	2/2	42.54	29.01	0.24
QDrop [44]		51.14	54.74	8.46
QDrop* [44]		54.72	58.67	13.05
PD-Quant [23]		53.14	57.16	13.76
Genie-M [15]		53.71	56.71	16.25‡
Bit-Shrinking* [22]		57.33	59.03	18.23
MetaAug (Ours)		54.22	57.30	16.97
MetaAug* (Ours)		**57.89**	**60.50**	**19.61**

16.90%), our approach achieves significant improvements of 3.08% and 1.35% over QDrop [44] in the test set for the 2/2 and 2/4 settings, respectively.

Table 3: Top-1 classification accuracy (%) of ResNet18 on 1024 calibration images (train set), and testing images of the ImageNet dataset, and the gap between accuracy on the calibration set and the test set.

Method	Bit-width (W/A)	Accuracy on the test set	Accuracy on the calibration set	Train-test accuracy gap
FP	32/32	71.01	85.16	14.15
QDrop [44]	2/2	51.14	77.53	26.39
PD-Quant [23]		53.14	83.30	30.16
Genie-M [15]		53.77	80.18	27.01
MetaAug (Ours)		54.22	77.64	**23.42**
QDrop [44]	2/4	64.66	81.64	16.98
PD-Quant [23]		65.17	84.38	19.21
Genie-M [15]		65.77	84.18	18.41
MetaAug (Ours)		66.01	82.91	**16.90**

Fig. 1: Visualization of the original calibration images (the first row) and the corresponding modified images (the second row) produced by the transformation network.

Visualization. Some original calibration images and the corresponding images produced by the transformation network are presented in Fig. 1. The images are produced with the setting 2/2 with the ResNet18 model. As shown in Fig. 1, the modified images change the appearance while still preserving the semantic information of the original calibration images.

4.4 Additional results

Comparisons with other augmentation approaches. We compare the results of our proposed method with various augmentation strategies, including traditional photometric data augmentation, such as contrast and brightness adjustments, and geometric data augmentation, such as random flipping and random rotation. We also investigate advanced augmentation methods, i.e.,

Table 4: Comparative Top-1 classification accuracy (%) with the 2/2 setting with ResNet-18 between our method and other augmentation approaches.

Augmentation	**Bit-width** (W/A)	**ResNet-18** (FP: 71.01)
Genie-M (no augmentation) [15]	2/2	53.71
Contrast		53.57
Brightness		53.35
Random Flip		53.93
Random Rotation		53.95
Random flip + Rotation + Brightness		53.87
Mixup [49]		54.05
Cutmix [48]		54.15
MetaAug (Ours)		54.22
MetaAug (Ours) + Mixup		54.35
MetaAug (Ours) + Cutmix		**54.63**

Mixup [49] and Cutmix [48]. The results of these augmentation techniques are presented in Table 4. The results show that the considered geometric augmentation strategies improve the performance over the baseline Genie-M [15], while an opposite observation is with the considered photometric. The results also show that our proposed method outperforms all compared augmentation strategies, including the advanced augmentation methods Mixup [49] and Cutmix [48]. This confirms the effectiveness of the proposed method. Furthermore, combining Mixup [49] or Cutmix [48] augmentation with our proposed method yields even more improvements. This indicates that our approach and existing advanced augmentation techniques can complement each other when used together.

5 Conclusion

In this paper, we propose a novel meta-learning based approach to mitigate the overfitting problem in post-training quantization. Specifically, we jointly optimize a transformation network, which is used to modify the original calibration data, and a quantized model in a bi-level optimization process. Additionally, we explore different losses, including an advanced distribution preservation loss, and propose using a margin loss for training transformation network so that the outputs of the network preserve the feature information of the original calibration data while preventing it from becoming an identity mapping. We extensively evaluate our proposed approach on the ImageNet dataset across various network architectures, demonstrating that the proposed method outperforms current state-of-the-art PTQ methods. A limitation of the current work is that the transformation network does not perform geometric transformations. In future work, we can consider designing a transformation network that also encodes geometric transformations, e.g., by integrating the Spatial Transformer module [14] to spatially transform regions of images. This will result in more diverse augmented images, which could improve the effectiveness of the proposed approach.

Acknowledgements

Trung Le and Dinh Phung were supported by ARC DP23 grant DP230101176 and by the Air Force Office of Scientific Research under award number FA2386-23-1-4044.

References

1. Abbas, M., Xiao, Q.W., Chen, L., Chen, P.Y., Chen, T.: Sharp-MAML: Sharpness-aware model-agnostic meta learning. In: ICML (2022)
2. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: NIPS. vol. 29 (2016)
3. Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Zeroq: A novel zero shot quantization framework. In: CVPR (2020)
4. Chen, S., Wang, W., Pan, S.J.: Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. NeurIPS (2019)
5. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: NIPS. pp. 3123–3131 (2015)
6. Défossez, A., Adi, Y., Synnaeve, G.: Differentiable model compression via pseudo quantization noise. TMLR (2022)
7. Duan, Y., Schulman, J., Chen, X., Bartlett, P.L., Sutskever, I., Abbeel, P.: Rl: Fast reinforcement learning via slow reinforcement learning. ArXiv (2016)
8. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned Step Size Quantization. In: ICLR (2020)
9. Fan, C., Ram, P., Liu, S.: Sign-MAML: Efficient model-agnostic meta-learning by SignSGD. ArXiv (2021)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017)
11. Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In: ICCV (2019)
12. Han, S., Mao, H., Dally, W.J.: Deep Compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: ICLR (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
14. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. NeurIPS (2015)
15. Jeon, Y., Lee, C., Kim, H.y.: Genie: Show me the data for quantization. In: CVPR (2023)
16. Jia, J., Feng, X., Yu, H.: Few-shot classification via efficient meta-learning with hybrid optimization. Engineering Applications of Artificial Intelligence (2024)
17. Kim, H.B., Lee, J.H., Yoo, S., Kim, H.S.: MetaMix: Meta-state precision searcher for mixed-precision activation quantization. In: AAAI (2024)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
19. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop (2015)

20. Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., Gu, S.: BRECQ: Pushing the limit of post-training quantization by block reconstruction. In: ICLR (2021)
21. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
22. Lin, C., Peng, B., Li, Z., Tan, W., Ren, Y., Xiao, J., Pu, S.: Bit-Shrinking: Limiting instantaneous sharpness for improving post-training quantization. In: CVPR (2023)
23. Liu, J., Niu, L., Yuan, Z., Yang, D., Wang, X., Liu, W.: Pd-quant: Post-training quantization based on prediction difference metric. In: CVPR (2023)
24. Ma, Y., Li, H., Zheng, X., Xiao, X., Wang, R., Wen, S., Pan, X., Chao, F., Ji, R.: Solving oscillation problem in post-training quantization through a theoretical perspective. In: CVPR (2023)
25. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research (2008)
26. Nagel, M., Amjad, R.A., Van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. In: ICML (2020)
27. Nagel, M., Baalen, M.v., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. In: CVPR (2019)
28. Nahshan, Y., Chmiel, B., Baskin, C., Zheltonozhskii, E., Banner, R., Bronstein, A.M., Mendelson, A.: Loss aware post-training quantization. Machine Learning **110**(11-12), 3245–3262 (2021)
29. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. ArXiv (2018)
30. Passalis, N., Tefas, A.: Learning deep representations with probabilistic knowledge transfer. In: ECCV (2018)
31. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: Imagenet classification using binary convolutional neural networks. In: ECCV (2016)
32. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR (2017)
33. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
34. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Li, F.F.: ImageNet large scale visual recognition challenge. IJCV (2015)
35. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
36. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: ICML (2016)
37. Satorras, V.G., Bruna, J.: Few-shot learning with graph neural networks. In: ICLR (2018)
38. Shin, J., So, J., Park, S., Kang, S., Yoo, S., Park, E.: Nipq: Noise proxy-based integrated pseudo-quantization. In: CVPR (2023)
39. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: NeurIPS (2017)
40. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H.S., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. CVPR (2017)
41. Vinyals, O., Blundell, C., Lillicrap, T.P., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NeurIPS (2016)

42. Wang, J.X., Kurth-Nelson, Z., Soyer, H., Leibo, J.Z., Tirumala, D., Munos, R., Blundell, C., Kumaran, D., Botvinick, M.M.: Learning to reinforcement learn. ArXiv (2016)
43. Wang, T., Wang, J., Xu, C., Xue, C.: Automatic low-bit hybrid quantization of neural networks through meta learning. ArXiv (2020)
44. Wei, X., Gong, R., Li, Y., Liu, X., Yu, F.: QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In: ICLR (2022)
45. Xu, S., Li, H., Zhuang, B., Liu, J., Cao, J., Liang, C., Tan, M.: Generative low-bitwidth data free quantization. In: ECCV (2020)
46. Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.S.: Quantization networks. In: CVPR (2019)
47. Youn, J., Song, J., Kim, H.S., Bahk, S.: Bitwidth-adaptive quantization-aware neural network training: A meta-learning approach. In: ECCV (2022)
48. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.J.: CutMix: Regularization strategy to train strong classifiers with localizable features. ICCV (2019)
49. Zhang, H., Cissé, M., Dauphin, Y., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
50. Zheng, D., Liu, Y., Li, L.: Leveraging inter-layer dependency for post-training quantization. NeurIPS (2022)