RepQ: Generalizing Quantization-Aware Training for Re-Parametrized Architectures

Anastasiia Prutianova[1]
auprutyanova@gmail.com
Alexey Zaytsev[2]
Likzet@gmail.com
Chung-Kuei Lee[1]
lee.chung.kuei@hisilicon.com
Fengyu Sun[1]
sunfengyu@hisilicon.com
Ivan Koryakovskiy[1]
i.koryakovskiy@gmail.com

[1] Huawei Technologies Co. Ltd.
Shanghai, China

[2] Beijing Institute of Mathematical Sciences and Applications
Beijing, China

Abstract

Existing neural networks are memory-consuming and computationally intensive, making deploying them challenging in resource-constrained environments. However, there are various methods to improve their efficiency. Two such methods are quantization, a well-known approach for network compression, and re-parametrization, an emerging technique designed to improve model performance. Although both techniques have been studied individually, there has been limited research on their simultaneous application. To address this gap, we propose a novel approach called RepQ, which applies quantization to re-parametrized networks. Our method is based on the insight that the test stage weights of an arbitrary re-parametrized layer can be presented as a differentiable function of trainable parameters. We enable quantization-aware training by applying quantization on top of this function. RepQ generalizes well to various re-parametrized models and outperforms the baseline method LSQ quantization scheme in all experiments.

1 Introduction

The number of parameters in Neural Networks (NN) has been growing over the years. This substantial computational complexity precludes the deployment of real-life NN-based applications to resource-constraint devices, e.g., mobile phones. Many research works aim at designing computationally efficient NN. A non-exhaustive list of ideas in this area includes knowledge distillation [26], model pruning [24], matrix factorization [12, 31], neural architecture search [37, 57], quantization [10, 21, 35], and re-parametrization [16, 52]. Here, we focus on re-parametrization and quantization as our main research fields.

Re-parametrization is an emerging technique that recently allowed the training of a plain non-residual VGG [16] model to achieve the remarkable accuracy of 80% on ImageNet [44] while being faster than ResNet-101 [25]. Moreover, re-parametrization has set a new state-of-the-art (SOTA) in channel pruning [14] and became a part of YOLO-7 [48]. The idea behind re-parametrization is that a neural architecture can be represented in different mathematical forms. Similar to how 2-D vectors on a plane can be defined by their Cartesian or polar coordinates, the same architecture can be expressed using various algebraic representations. An alternative representation helps gradient descent to reach a better local minimum, resulting in improved performance. In practical applications, re-parametrization involves replacing linear layers, such as fully-connected and convolutional layers, with a combination of linear layers. For example, several papers [13, 16, 23, 27, 52] have proposed replacing each convolution with a block consisting of multiple convolutional layers with various kernel sizes, channel numbers, residual connections, and batch normalization layers. This re-parametrization block is used during training but is equivalently converted back into a single convolution during inference. To sum up, boosting model quality without additional computational burden at inference makes re-parametrization an important method for increasing NN efficiency. Efficiency is a feature of another major compression technique, namely quantization. Basic quantization algorithms can achieve more than a 75% reduction in model size while maintaining a performance comparable to uncompressed models. This makes quantization a key technique for real-life model deployment.

We aim to bring the advances in re-parametrization research to practical applications by providing a suitable approach to quantizing re-parametrized NN. Currently, only two research works consider quantizing re-parametrized NN [11, 17], and they both target a particular architecture, RepVGG. Although novel, the obtained results show a considerable quality drop due to quantization which eliminates the improvements introduced by the re-parametrization.

To solve the problem of insufficient quality and generalization, we introduce a Quantization-Aware Training (QAT) strategy for re-parametrized NN. The challenge is that the training stage weights differ from the testing stage weights, prohibiting standard QAT application [21]. In particular, applying a regular quantization independently to each convolution of a re-parametrized block prevents its merging during inference without increasing the quantizer bit width. For instance, two sequential convolutions of bit-width two will merge into a single convolution of bit-width four. Instead, we propose to compute the inference stage parameters of a convolution as a differentiable function of the trainable parameters of the re-parametrized block and then apply a pseudo-quantization function on top of it. This way, our RepQ approach enables end-to-end quantization-aware re-parametrized training.

To sum up, our main contributions are the following.

- We are the first to propose a method that allows QAT of models with arbitrary re-parametrization schemes. We provide extensive experiments showing that our RepQ method of joint quantization with re-parametrization leads to consistent quality enhancement on SOTA architectures. In particular, for the first time, we achieved lossless 8-bit quantization of re-parametrized models.

- Batch Normalization (BN) contained in the re-parametrized blocks creates a challenge for differentiable merging and hence for QAT. We show how to compute differentiable weight functions via BN-folding. Lastly, we enhance BN-folding by introducing BN statistics estimation, which reduces the computational complexity in theory and practice.

2 Related works

Re-parametrization. Early re-parametrizations appeared as a result of batch normalization [29] and residual connection [25] research. Inspired by batch normalization, the authors of [45] proposed a weights normalization technique. This technique can be viewed as a re-parametrization that decouples the direction and length of the weights vector by introducing an additional parameter responsible for the weight norm. The authors of DiracNets [52] looked for a way to train deep networks without explicit residual connections. They proposed re-parametrizing a convolution with a convolution combined with a skip connection. It allowed very deep single-branch architectures to reach a decent quality. ACNets [13] and RepVGG [16] brought re-parametrization to a new level by introducing multiple convolutions and batch normalization to re-parametrization blocks, revisiting well-known architectures like VGG [46] and ResNet [25] and significantly enhancing their quality. More advanced [15, 27] re-parametrization strategies tend to use a larger number of convolutions and more diverse branches in their blocks. Besides its extensive application to classification tasks, re-parametrization recently helped to reduce computational burden in other computer vision tasks like object detection [34, 48] and super-resolution [49, 54]. Re-parametrization receives theoretical justification in [3]; the authors prove that under certain theoretical assumptions in a simple convex problem, re-parametrization leads to faster convergence. The authors of [17] show that in some cases, re-parametrization can be equivalent to regular network training with a certain gradient scaling strategy.

Quantization. Regularly, NN parameters are stored as floating point numbers. Yet, a 32-bit parameter representation is redundant to maintaining the network's quality. The research field aiming to find the optimal low-bit integer parameters' representation is known as neural network quantization. Theoretically, quantization involves rounding, leading to zero gradients in the network almost everywhere. QAT algorithms are used to address this issue by simulating quantization in a differentiable manner, allowing the network to adapt for subsequent quantization. For instance, [5, 20] suggested injecting pseudo-quantization noise into full-precision network training. Alternatively, [10, 21, 33] uses a straight-through estimator [6] to approximate the gradients of the discontinuous quantization function on the backward pass. [20, 22, 43] proposes a smooth approximation of this stair-step function. In addition, the listed base methods could be improved by using knowledge distillation [42], progressive quantization [56], stochastic precision [18], Batch Normalization re-estimation [39], additional regularization [2, 36], and various non-uniform quantization extensions [19, 38, 51].

3 Background

Quantization allows for reducing inference time and power consumption by decreasing the precision of matrix multiplication factors in convolutions or fully-connected layers. Quantization can result in quality degradation, so QAT is applied to recover the quality and ensure model resilience. During QAT, the original convolution operation, denoted as $X * W$, with input X and weight W, is transformed into $Q(X) * Q(W)$, where Q is a pseudo-quantization function that allows back-propagation and $*$ represents a convolutional operator. LSQ [21] is the current SOTA pseudo-quantization function, so we also use it in our experiments.

When quantizing re-parameterized models, there are several options available.

1. Apply re-parametrization and Post-Training Quantization (PTQ) [4, 9, 41] successively. Research works [11, 17] show that re-parametrization can lead to PTQ-un-

friendly distributions, resulting in a significant quality drop after PTQ application.

2. Apply re-parametrization and QAT successively. Trained re-parametrized full-precision blocks are converted into single layers, and standard QAT is applied to each of them.

3. Apply re-parametrization and QAT simultaneously as follows. Quantize each layer inside a re-parametrized block independently and then merge those layers into a single quantized layer only after QAT. This option is impractical as merging quantized convolutions results in a convolution with a higher bit width while binary and lower bit quantization is much more challenging than four or eight-bit quantization [50]. Consider a simple multiplication function $f(x) = xw$, which is re-parametrized as follows $R(x) = xw_1w_2$, with the resulting merged weight $w = w_1w_2$. When w_1 and w_2 are in FP32, their multiplication usually incurs a negligible loss of precision. However, if w_1 and w_2 are quantized to 2 bits, represented by integers in the range [0, 3], their multiplication results in an integer in the range [0, 9], requiring at least 4 bits for storage.

4. We introduce a novel approach where QAT and re-parametrization are applied simultaneously by performing pseudo-quantization on top of the merged re-parametrized block.

Options 1-3 are based on well-known approaches to regular model quantization, while option 4 is our novel approach designed specifically for the quantization of re-parametrized models. We exclude options 1 and 3 as they are either not generic or known to produce unsatisfactory results for the re-parametrized models. In our experiments, we compare option 2 to our proposed RepQ method (option 4) to demonstrate its effectiveness.

4 Proposed Methods

In Section 4.1, we describe the quantization strategy of re-parameterized blocks without BN layers and introduce a general RepQ training framework. For blocks with BN layers, we provide two alternative extensions in Sections 4.2 and 4.3.

4.1 RepQ: Quantization-aware training with re-parametrization

Let us first consider the scenario in which BN is not used in a re-parametrized block. The authors of [27] notice that it is possible to reduce training time by merging the re-parametrized blocks without BN into a single convolution while still optimizing the extended set of weights introduced by re-parametrization. This section shows how this merged training benefits QAT.

To illustrate the concept of re-parametrization, we use the simple example shown in Figure 1: $R(X,W) = X * W_1 * W_2 + X * W_3$. $R(X,W)$ denotes a re-parametrized block that replaces a single convolution during training. We can simplify this block to a single convolution with weight M by deducing that $R(X,W) = X * (W_1 * W_2 + W_3) = X * M(W_1, W_2, W_3)$. In a broader sense, M is a differentiable function that maps the block's trainable parameters to the weight of the final converted convolution. This example easily generalizes to the other re-parametrization strategies, where the re-parametrized block has a form of $R(X,W) = X * M(W_1, \dots, W_n)$ or, roughly speaking, re-parametrizations without BN. The articles introducing novel re-parametrized blocks regularly provide the formulas necessary to compute M, so we do not repeat them here.

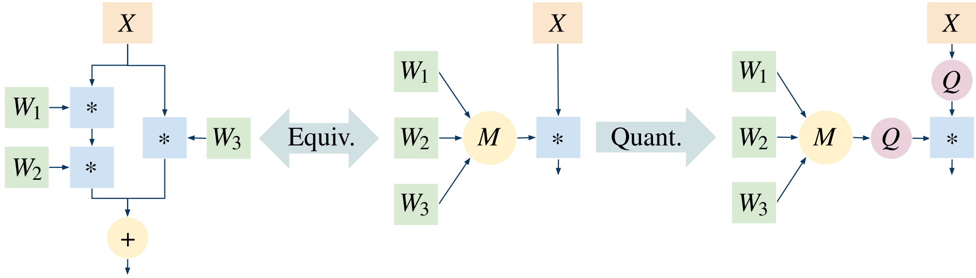

Figure 1: This picture illustrates the application of a re-parametrized quantization-aware training (RepQ) to a single layer with convolutions with input X and parameters W_1, W_2, W_3. The left plot illustrates the re-parametrization block substituting a single convolutional layer used as an example. The middle plot is an equivalent transformation (in terms of the gradient flow) to the left plot. Note that the computation order is different. The right plot shows how the pseudo-quantization functions Q are inserted to perform RepQ.

Notably, merged training does not affect the gradient flow, as the two sides of the equation are numerically equivalent during both forward and backward passes. However, the merged weight M is explicitly computed on the right side.

Now it is easy to see that the pseudo-quantization function can be applied on top of M:

$$X * M(W_1, \dots, W_n) \to Q(X) * Q(M(W_1, \dots, W_n)). \tag{1}$$

As a result, function $Q(M(W_1, \dots, W_n))$ will equal the quantized weight used on the inference. Since Q and M are differentiable functions, the gradient propagates smoothly to the weights $W_1, \dots, W_n$. The combination of re-parametrization with introducing M and pseudo-quantization function Q constitutes the RepQ approach and enables end-to-end quantization-aware training. This is the main algorithm used in all RepQ experiments.

4.2 RepQ-BN: Merging batch normalization

Many SOTA re-parametrizations use batch normalization. Several papers show that BN is an essential component of their blocks. BN's removal leads to a significant performance drop. Since we aim to provide a quantization strategy that generalizes well to diverse re-parametrizations, we study how to handle BN in QAT.

The first option is fusing the BN with the preceding convolution during training, described in this section. A similar procedure was proposed in [30, 32] to achieve integer-arithmetic-only quantization. In our case, folding BN reduces the task to the no-BN case described in the previous section, 4.1.

$$\begin{aligned} \mathcal{BN}(X*W) &= \frac{X*W - \mathbb{E}\left[X*W\right]}{\sqrt{\mathbb{V}\left[X*W\right]+\varepsilon}}\gamma + \beta = \\ &= X * \frac{W}{\sqrt{\mathbb{V}\left[X*W\right]+\varepsilon}}\gamma - \frac{\mathbb{E}\left[X*W\right]}{\sqrt{\mathbb{V}\left[X*W\right]+\varepsilon}}\gamma + \beta = X * M(X,W) + b(X,W). \end{aligned} \tag{2}$$

The equation (2) shows that a convolution followed by BN is equivalent to a single convolutional operator. However, its parameters are dependent on the input X through the batch statistics, mean and variance. Formally, re-parametrization with BN has the form of $R(X,W) = X * M(X, W_1, \dots, W_n)$. Algorithm 1 shows how to compute M and apply quantization in practice for a simple case of $R(X,W) = \mathcal{BN}(X*W)$. By fusing BN with preceding convolutions, we reduce the task of merging weights to the no-BN case described in Subsection 4.1. We call this variant of our approach RepQ-BN.

Algorithm 1 Fusing BN with Convolution for Quantization

Given: $R(X,W) = \mathcal{BN}(X*W)$ ▷ a simple re-parametrization of BN after a convolution

1: $Y = X * W$
2: $\mu, V = \mathbb{E}\Big[Y\Big], \mathbb{V}\Big[Y\Big]$ ▷ computing BatchNorm statistics
3: $\hat{\mu} = (1-m)\cdot\hat{\mu} + m\cdot\mu$ ▷ updating cumulative moving mean (m denotes momentum)
4: $\hat{V} = (1-m)\cdot\hat{V} + m\cdot V$ ▷ updating cumulative moving variance
5: $M(X,W) = \frac{W}{\sqrt{V+\varepsilon}}\cdot\gamma$ ▷ computing merged weight
6: $b(X,W) = -\frac{\mu}{\sqrt{V+\varepsilon}}\cdot\gamma + \beta$ ▷ computing merged bias
7: $R_q(X,W) = Q(X) * Q(M(X,W)) + b(X,W)$ ▷ quantized re-parametrized layer

4.3 RepQ-BNEst: Batch normalization estimation

An observant reader may have noticed that in Algorithm 1, the convolutional operator is computed twice, first in line 1 and then again in line 7. While the additional computation in line 1 is used to calculate the BN statistics μ and V, the question arises whether it is necessary to perform such a computationally expensive convolution to determine the mean and variance of the output. Here we propose a novel method of estimating BN running statistics based on inputs and weights without computing the convolution.

For simplicity, let us consider the case of a 1×1 convolution. The 1×1 convolution can be viewed as a matrix multiplication of the input X of the shape $[B\cdot H\cdot D, IN]$ and weight W of shape $[IN, OUT]$, where B is the batch size, H is the image height, D is the image width, IN is the number of input channels, and OUT is the number of output channels. Consider computing BN mean statistics,

$$\mathbb{E}\left[XW\right] = \mathbb{E}\left[X\right]W. \tag{3}$$

The equation suggests we can first calculate the per-channel mean over the batch, height, and width dimensions and then multiply the result by the weight matrix. This approach has a computational complexity of $\mathrm{O}(B\cdot H\cdot D\cdot IN)$, making it more favourable than the naive solution, which has a complexity of $\mathrm{O}(B\cdot H\cdot D\cdot IN\cdot OUT)$. Additionally, BN estimation allows us to avoid storing the feature map XW on the GPU.

For exact variance computation, a similar reduction is not possible due to the need to calculate the input covariance matrix. As a solution, we propose to approximate the covariance matrix with a diagonal form,

$$\mathbb{V}\Big[XW\Big] = \mathcal{D}\Big[W^T\mathrm{Cov}(X,X)W\Big] \approx \mathcal{D}\Big[W^T\mathcal{D}\Big[\mathrm{Cov}(X,X)\Big]W\Big] = \mathbb{V}\Big[X\Big]W^2, \tag{4}$$

where W^2 is an element-wise square of W, Cov is the sample covariance matrix, and $\mathcal{D}$ is the diagonalizing operator that leaves only diagonal matrix elements non-zero. As a result of BN estimation, the variance is substituted with another quadratic statistic of the weight and the output that estimates variance but is computationally more efficient for quantization-aware re-parametrization. The above formulas generalize to arbitrary weight shapes as follows (for more details, we refer to supplementary materials),

$$\mathbb{E}\Big[X * W\Big] \approx \tilde{\mathbb{E}}\Big[X * W\Big] = \mathbb{E}\Big[X\Big] \cdot \sum_{h,d} W_{h,d}, \tag{5}$$

$$\mathbb{V}\Big[X * W\Big] \approx \tilde{\mathbb{V}}\Big[X * W\Big] = \mathbb{V}\Big[X\Big] \cdot \sum_{h,d} W_{h,d}^2. \tag{6}$$

To sum up, for BN Estimation, we modify Algorithm 1 by replacing the calculation of mean μ and variance V with $\tilde{\mathbb{E}}$ and $\tilde{\mathbb{V}}$, respectively, in line 2. In addition, we skip computing Y in line 1. We call this variant of our approach RepQ-BNEst.

5 Experiments

5.1 Experimental setup

Architectures. We evaluate the performance of RepQ on three architectures: ResNet-18, VGG, and ECBSR [54]. For ResNet-18, we employ two re-parametrization techniques, the well-known ACNets [13] and the recently published OREPA [27], which we refer to as AC-ResNet-18 and OREPA-ResNet-18, respectively. For VGG, we use the SOTA RepVGG [16] approach to re-parametrization with two network depth variants, RepVGG-A0 and RepVGG-B0. The ECBSR is a re-parametrization approach developed for the super-resolution problem. Our focus on evaluating RepQ with lightweight versions of SOTA architectures was motivated by their suitability for mobile devices.

Comparisons. We compare our RepVGG quantization results with QARepVGG, a quantization-friendly version of VGG introduced in [11]. To the best of our knowledge, no studies provide quantization results for re-parametrized architectures apart from RepVGG. To further support the main claims of our article, we have designed baselines, which are described in the following paragraph.

Baselines. Quantized model training includes two sequential stages: (1) a regular full-precision (FP) pre-training and (2) QAT. Weights pre-trained on the FP stage are used to initialize the quantized model at the beginning of the second stage. We compare our re-parametrized quantization-aware training RepQ that uses re-parametrization during QAT with the baselines that do not use re-parametrization in the QAT stage. In particular, we compare RepQ to plain and merged baselines schematically described in Table 1:

- **Plain.** Regularly trained model with no re-parametrization in both stages.

- **Merged.** The re-parametrized model is trained in the FP stage. Re-parametrized blocks are merged back into single convolutions, and the merged weights are used to initialize the quantized model. Such initialization can be profitable for subsequent quantization because re-parametrized models usually have better metrics than regular models.

	FP pre-training stage		QAT stage	
	Regular	Re-parametrized	Regular	Re-parametrized
Plain	✓		✓	
Merged		✓	✓	
RepQ		✓		✓

Table 1: Quantization strategies: baselines (Plain, Merged) and our proposed method(RepQ).

Training pipeline. Generally, re-parametrization changes only the model architecture. For example, imagine that all convolutions are replaced by the following re-parametrization blocks $R(X) = BN(X * W) + X$. The simplified pseudo-code of this block is shown below.

```
class ReparametrizedBlock:
  def kernel(x, training):
    # Proposed differentiable weight function, M.
    BN_stats = self.BN_stats(x, self.conv.weight, training)
    weight = fuse_BN(self.conv.weight, BN_stats)
    return fuse_residual(weight)

  def BN_stats(x, weight, training):
    if self.BNEst:
      return BNEst(x, weight, training) # Use Eq. 5-6.
    else:
      return  BN(x, weight, training) # Use Algorithm 1..

  def call(x, training=False):
    # "training" defines the training or testing mode.
    if self.quantize:
      return conv2d(Q(x), Q(self.kernel(x, training))) # Quantized training.
    elif self.BNEst:
      return conv2d(x, self.kernel(x, training)) # BN estimation for FP training.
    else:
      return self.bn(self.conv(x)) + x   # Regular FP training with BN.
```

Once full-precision training converges, the "self.quantize" parameters are set to True, and training is repeated with minor changes to hyperparameters, described in the supplementary.

Implementation details. We trained full-precision models in accordance with the re-parametrization articles' setup using official repositories. That is why the baseline for the same model architecture may slightly differ for various re-parametrization blocks. We trained quantized models with the same hyperparameter setup as the full-precision models, except for learning rate adjustments. All quantized models are initialized with corresponding pre-trained full-precision weights; quantization steps are initialized using MinError [8] initialization for the first batch. For reproducibility, we provide hyperparameters in the supplementary materials.

5.2 Experimental results

We present our main results in Tables 2 and 3. They demonstrate that the proposed variants of RepQ consistently outperform the baselines of all the tested architectures and re-parametrized blocks. Table 3 exhibits the benefits of RepQ on a wide range of super-resolution datasets. For all the studied classification models, the 8-bit RepQ performance exceeds the full-precision result, while the QARepVGG [11] exhibits a quality drop. The gap between our RepQ and QARepVGG exceeds 1% for both RepVGG-A0 and RepVGG-B0. For RepVGG and ResNet, the best result is achieved either by RepQ-BN or RepQ-BNEst. In particular, RepQ-BN is slightly better than RepQ-BNEst in eight-bit experiments; nevertheless, RepQ-BNEst achieves better results in four-bit quantization. Surprisingly, RepQ-BNEst outperforms RepQ-BN by a considerable margin on the 4-bit RepVGG-A0 and RepVGG-B0. In addition, we measured the ResNet-18 training speed on 2 V100 GPUs. RepQ-BNEst allows for a **25%** training time reduction compared to RepQ-BN. The merged baseline demonstrates inconsistency on different studied architectures. For RepVGG-B0, the merged results are quite close to RepQ; however, for Resnet-18 and ECBSR, it is inferior to plain models.

Model	Method	Precision 32 (FP)	8	4
RepVGG-A0 [16]	QARepVGG [11]	<u>72.40</u>	71.90	-
	Plain	70.91	71.52	69.90
	Merged	-	72.49	69.21
	RepQ-BN	72.25	**73.11**	<u>70.31</u>
	RepQ-BNEst	**72.43**	<u>72.94</u>	**71.12**
RepVGG-B0 [16]	QARepVGG [11]	75.10	74.60	-
	Plain	73.48	74.79	72.52
	Merged	-	75.48	73.06
	RepQ-BN	<u>75.27</u>	**75.60**	<u>73.51</u>
	RepQ-BNEst	**75.38**	<u>75.53</u>	**74.01**
AC-Resnet-18 [13]	Plain	70.24	70.77	70.00
	Merged	-	70.64	68.25
	RepQ-BN	**71.01**	**71.40**	<u>70.07</u>
	RepQ-BNEst	<u>70.80</u>	<u>71.23</u>	**70.20**
OREPA-Resnet-18 [27]	Plain	<u>71.24</u>	<u>71.76</u>	**71.49**
	Merged	-	69.58	69.96
	RepQ	**72.07**	**72.32**	**71.49**

Table 2: ImageNet [44] top-1 accuracy for different quantization strategies for re-parametrized NNs. The best result is emphasized with bold text, and the second best is underscored.

Precision	Method	Set5 [7]	Set14 [53]	B100 [40]	U100 [28]
32		36.90	32.62	31.43	29.66
8	Plain	36.81	32.55	31.39	29.57
	Merged	36.86	32.54	31.39	29.53
	RepQ	**36.91**	**32.57**	**31.40**	**29.60**
4	Plain	36.20	32.16	**31.09**	28.91
	Merged	36.16	32.15	31.08	**28.95**
	RepQ	**36.30**	**32.19**	**31.09**	**28.95**
2	Plain	34.82	31.30	30.46	27.92
	Merged	34.74	31.28	30.44	27.91
	RepQ	**34.89**	**31.36**	**30.50**	**28.00**

Table 3: PSNR metric for ECBSR-m4c8 [54] architecture trained on DIV2K [1] dataset, and x2 scaling.

5.3 Discussion

Our results demonstrate that 8-bit quantization can improve the quality of re-parametrized models. A similar behaviour is also observed in plain models [21, 47, 55] and usually explained by an additional regularization effect of quantization. Four-bit quantization with the RepQ method results in a minor quality drop. At the same time, the number of bit-operations is reduced by four times compared to 8-bit models. Interestingly, 4-bit RepVGG-B0 has two times less bit-operations than 8-bit RepVGG-A0, while its accuracy is higher. This makes 4-bit RepVGG-B0 more favourable for deployment.

Limitations. The main limitation of re-parametrization and RepQ is the increase in training time (TT). Let us take as an example ResNet-18 re-parametrized with two different blocks: ACNets and OREPA. The table provides the relative training time of the re-parametrized models for Plain FP and QAT networks. The TT increase introduced by RepQ is comparable to the one introduced by re-parametrization on full-precision training for both ACNets and OREPA blocks. Although RepQ-BN experiences a longer training due to the twice-forward computation, RepQ-BNEst mitigates this issue. Overall, despite the training time overhead, re-parametrization and RepQ is beneficial when the trade-off between inference time and model quality is the priority.

Block	FP		QAT			
	Plain	Re-parametrized	Plain	RepQ	RepQ-BN	RepQ-BNEst
ACNets	100%	200%	100%	-	195%	150%
OREPA	100%	210%	100%	165%	-	-

Table 4: Training time overhead for different re-parametrization strategies on ResNet-18.

6 Conclusions

This paper introduces RepQ, a QAT strategy specifically designed for re-parametrized models. During training, RepQ merges re-parametrized blocks into a single convolution and applies a pseudo-quantization function on top of the merged weight. To be able to quantize arbitrary re-parametrization blocks, we provide a natural solution for merging non-linear batch normalization layers inside the blocks. Furthermore, we enhance this solution by estimating BN statistics and thus achieve a speed-up. We conduct an extensive experimental evaluation for diverse re-parametrized blocks and model architectures. Results show that RepQ surpasses existing solutions and provides lossless 8-bit quantization for the re-parametrized models. Overall, RepQ expands the applicability of re-parametrization to the field of quantized NN with an easy-to-implement approach.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[2] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient l1 Regularization for Quantization Robust-

ness. In *Proceedings of the International Conference on Learning Representations*, 2020.

[3] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. In *Proceedings of The International Conference on Machine Learning*, 2018.

[4] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Proceedings of the Conference on Neural Information Processing Systems*, 2018.

[5] Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M. Bronstein. NICE: Noise Injection and Clamping Estimation for Neural Network Quantization. In *Proceedings of the International Conference on Learning Representations*, 2018.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432*, 2013.

[7] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-line Alberi Morel. Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. In *Proceedings of the British Machine Vision Conference*, 2012.

[8] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. LSQ+: Improving low-bit quantization through learnable offsets and better initialization. In *Camera-ready for Joint Workshop on Efficient Deep Learning in Computer Vision, CVPR*, 2020.

[9] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[10] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv:1805.06085*, 2018.

[11] Xiangxiang Chu, Liang Li, and Bo Zhang. Make RepVGG Greater Again: A Quantization-aware Approach. *arXiv:2212.01593*, 2022.

[12] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting Parameters in Deep Learning. In *Proceedings of the Conference on Neural Information Processing Systems*, 2013.

[13] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. ACNet: Strengthening the Kernel Skeletons for Powerful CNN via Asymmetric Convolution Blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[14] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. ResRep: Lossless CNN Pruning via Decoupling Remembering and Forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[15] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse Branch Block: Building a Convolution as an Inception-like Unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[16] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[17] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing Your Optimizers rather than Architectures. In *Proceedings of the International Conference on Learning Representations*, 2023.

[18] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Jun Zhu, and Hang Su. Learning accurate low-bit deep neural networks with stochastic quantization. In *Proceedings of the British Machine Vision Conference*, 2017.

[19] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. In *Proceedings of the Conference on Neural Information Processing Systems*, 2020.

[20] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable Model Compression via Pseudo Quantization Noise. *Transactions on Machine Learning Research*, 2022.

[21] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. LEARNED STEP SIZE QUANTIZATION. In *Proceedings of the International Conference on Learning Representations*, 2020.

[22] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[23] Shuxuan Guo, Jose M. Alvarez, and Mathieu Salzmann. ExpandNets: Linear Overparameterization to Train Compact Convolutional Networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Proceedings of the Conference on Neural Information Processing Systems*, 2020.

[24] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both Weights and Connections for Efficient Neural Network. In *Proceedings of the Conference on Neural Information Processing Systems*, 2015.

[25] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.

[26] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[27] Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiansheng Hua, and Xiaojin Gong. Online Convolutional Re-parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[28] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.

[29] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of The International Conference on Machine Learning*, 2015.

[30] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[31] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. In *Proceedings of the British Machine Vision Conference*, 2014.

[32] Qing Jin, Jian Ren, Richard Zhuang, Sumant Hanumante, Zhengang Li, Zhiyu Chen, Yanzhi Wang, Kaiyuan Yang, and Sergey Tulyakov. F8net: Fixed-Point 8-bit Only Multiplication for Network Quantization. 2021.

[33] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Youngjun Kwak, Jae-Joon Han, and Changkyu Choi. Joint training of low-precision neural network with quantization interval parameters. *arXiv:1808.05779*, 2018.

[34] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, and others. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv:2209.02976*, 2022.

[35] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv:1605.04711*, 2016.

[36] Ji Lin, Chuang Gan, and Song Han. Defensive Quantization: When Efficiency Meets Robustness. In *Proceedings of the International Conference on Learning Representations*, 2019.

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *Proceedings of the International Conference on Learning Representations*, 2019.

[38] Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[39] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed Quantization for Discretized Neural Networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

[40] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, 2001.

[41] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or Down? Adaptive Rounding for Post-Training Quantization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the International Conference on Machine Learning*, 2020.

[42] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *Proceedings of the International Conference on Learning Representations*, 2018.

[43] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2014.

[45] Tim Salimans and Diederik P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Proceedings of the Conference on Neural Information Processing Systems*, 2016.

[46] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations*, 2014.

[47] Kirill Solodskikh, Vladimir Chikin, Ruslan Aydarkhanov, Dehua Song, Irina Zhelavskaya, and Jiansheng Wei. Towards Accurate Network Quantization with Equivalent Smooth Regularizer. In *Proceedings of the European Conference on Computer Vision*, 2022.

[48] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv: 2207.02696*, 2022.

[49] Xintao Wang, Chao Dong, and Ying Shan. RepSR: Training Efficient VGG-style Super-Resolution Networks with Structural Re-Parameterization and Batch Normalization. In *Proceedings of the International Conference on Multimedia*, 2022.

[50] Bin Xia, Yulun Zhang, Yitong Wang, Yapeng Tian, Wenming Yang, Radu Timofte, and Luc Van Gool. Basic Binary Convolution Unit for Binarized Image Restoration Network. *Proceedings of the International Conference on Learning Representations*, 2023.

[51] Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[52] Sergey Zagoruyko and Nikos Komodakis. DiracNets: Training Very Deep Neural Networks Without Skip-Connections. *arXiv:1706.0038*, 2017.

[53] Roman Zeyde, Michael Elad, and Matan Protter. On Single Image Scale-up Using Sparse-Representations. In *Proceedings of the International Conference on Curves and Surfaces*, 2010.

[54] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented Convolution Block for Real-time Super Resolution on Mobile Devices. In *Proceedings of the International Conference on Multimedia*, 2021.

[55] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160*, 2016.

[56] Bohan Zhuang, Mingkui Tan, Jing Liu, Lingqiao Liu, Ian Reid, and Chunhua Shen. Effective training of convolutional neural networks with low-bitwidth weights and activations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[57] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations*, 2017.