TRQ: Ternary Neural Networks With Residual Quantization

Yue Li[1], **Wenrui Ding**[1*], **Chunlei Liu**[1], **Baochang Zhang**[1], **Guodong Guo**[2]
[1] Beihang University
[2] Institute of Deep Learning, Baidu Research and National Engineering Laboratory for Deep Learning Technology and Application
ppy@buaa.edu.cn, ding@buaa.edu.cn, liuchunlei@buaa.edu.cn, bczhang@buaa.edu.cn, guoguodong01@baidu.com

Abstract

Ternary neural networks (TNNs) are potential for network acceleration by reducing the full-precision weights in network to ternary ones, *e.g.*, $\{-1, 0, 1\}$. However, existing TNNs are mostly calculated based on rule-of-thumb quantization methods by simply thresholding operations, which causes a significant accuracy loss. In this paper, we introduce a stem-residual framework which provides new insight into ternary quantization, termed Ternary Residual Quantization (TRQ), to achieve more powerful TNNs. Rather than directly thresholding operations, TRQ recursively performs quantization on full-precision weights for a refined reconstruction by combining the binarized stem and residual parts.With such a unique quantization process, TRQ endows the quantizer with high flexibility and precision. Furthermore, our TRQ is generic, which can be easily extended to multiple bits through recursively encoded residual for a better recognition accuracy. Extensive experimental results demonstrate that the proposed method yields great recognition accuracy while being accelerated.

Introduction

Low precision neural networks (Esser et al. 2019; Zhou et al. 2016; Jung et al. 2019; Yang et al. 2020) are currently emerging as important machine learning technologies, as the result of the growing need for compressing the best Deep Neural Networks (DNNs) to support embedded devices with limited storage and computing capabilities. Ternay Neural Networks (TNNs) (Alemdar et al. 2017; Laborieux et al. 2020; Zhu et al. 2016; Li, Zhang, and Liu 2016; Deng et al. 2018), which constrain the key data structures (weights and activations) to the ternary space $\{-1, 0, 1\}$, are one of the extreme cases in low precision neural networks. TNNs could directly replace the multiply-accumulate operations by controlling gate along with binary logical operations, *i.e.*, XNOR (Rastegari et al. 2016), thus drastically simplifying the consumption and reducing the memory cost in the inference phase of DNNs.

For TNNs, to alleviate the performance degradation brought by ternarization, a suitable quantizer that can accurately map full-precision values to the quantized ternary ones is absolutely important. Many approaches have been explored for quantizer optimization. TWN (Li, Zhang, and Liu 2016) first proposed a weight quantizer that performs ternarization via fixed thresholds and scale factors, consequently removing the multiplication operations and improving the resource-efficient. Followed by TWN, TTQ (Zhu et al. 2016) optimizes the weight quantizier by learning different scale factors for different states of weights, achieving higher performance on recognition tasks. Besides, GXNOR-Net (Deng et al. 2018) focuses on activition ternarization and employs a derivative approximation technique for backward optimization of quantizer. Actually, though the ternary quantizers have been improved in the above literatures, they are consistently trained with simple thresholding operations, which leads to large approximation error to full-precision weights and remains the significant performance gap.

In this paper, we aim at designing a more accurate and flexible ternary quantizer to improve the performance of TNNs. Specifically, we propose ternary neural networks with residual quantization (TRQ), whcih provides new insight into ternarization by introducing a stem-residual framework. As shown in Figure 1, rather than directly thresholding operations, we perform the ternary weights (O_1) by the combination of binarized stem and residual, which are obtained via recursive quantization on full-precision weights. Such stem-residual quantization brings a refined reconstruction for ternary weights, thus achieving small quantization error and accurate mapping. Specially, a learnable coefficient is introduced in the quantization process, which avoids a dedicated tuning and automatically searchs for a better approximation to the full-precision weights. Besides, TRQ could be extended to multiple bits by recursively encoding the residual, leading to a new method for multi-bit quantization. Experimental results indicate the TRQ has a superior performance on CIFAR-10/100 and ImageNet. We summarize our contributions as follows:

1) We propose Ternary Neural Networks with Residual Quantization (TRQ) in the stem-residual framework, which significantly improves the performance of the thresholding based methods by calculating ternary weights in a recursive manner.

2) Our method is based on a learnable quantization scale, and a more reasonable training process to better approximate full-precision weights.

3) We evaluate TRQ on three diverse classification

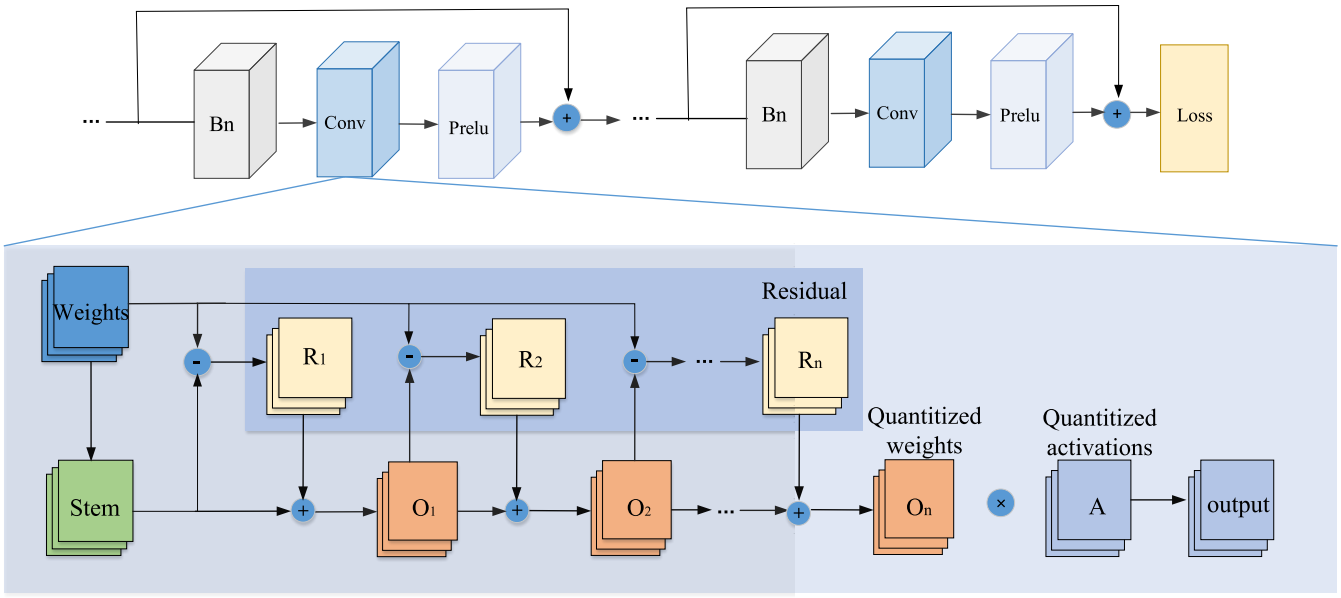


Figure 1: The stem-residual framework of ternary neural networks with residual quantization. As illustrated in the figure, TRQ recursively performs quantization on full-precision weights for a refined reconstruction by combining the binarized stem and residual parts. In the figure, R_n and O_n represent the residual and quantitized weights at the nth quantization level. When $n = 1$, the ternary quantizer is achieved.

datasets, demonstrating that our method performs much better than the state-of-the-art approaches.

Related Work

Binary Neural Network. Binary neural networks (BNNs), in which both activition and weights are quantizated to binary values, *e.g.*, $\{-1, 1\}$, are particularly adopted as a popular model acceleration way because all calculation can be realized through bit-wise operations (Rastegari et al. 2016) and avoid multiplications. As demonstrated in XNOR-Net (Rastegari et al. 2016), a binary convolution layer can achieve $32\times$ memory saving and $58\times$ speed-up on CPUs. However, the extreme compression rate also comes with a significant performance degradation. XNOR-Net based on a ResNet-18 (He et al. 2016) architecture only achieves 51.2% accuracy on the ImageNet classification dataset, leaving a 18% accuracy gap from its full-precision counterpart. Though series of approaches have been explored for training BNNs, including weight approximation (Bulat and Tzimiropoulos 2019; Gu et al. 2019a; Rastegari et al. 2016), architecture redesign (Liu et al. 2020a; Zhuang et al. 2019), loss reconstruction (Hou, Yao, and Kwok 2016; Gu et al. 2019b; Liu et al. 2020b) and training strategy optimization (Kim et al. 2020; Han et al. 2020), the performance gap issue is still unsolved.

Ternary Neural Network. TNNs constrain the full-precision values in the ternary space $\{-1, 0, 1\}$, resulting in an extra zero state compared with BNNs. However, TNNs can also be implemented as efficient as BNNs. As demonstrated in GXNOR-Nets (Deng et al. 2018), for a multiplication operation in TNNs, when one of the weight and activation is zero or both of them are zeros, the corresponding computation unit is resting, until the non-zero weight and non-zero activation enable and wake up the required computation unit. Thus (Deng et al. 2018) regards computation trigger determined by the weight and activation as a control signal/gate to start the XNOR computation and implement GXNOR-Nets in an event-driven paradigm. Further, (Laborieux et al. 2020) extends the hardware implementation of BNNs to TNNs on a hybrid 130 nm CMOS/RRAM chip, demonstrating that the same memory array architecture can be used to implement ternary weights instead of binary weights. Moreover, (Alemdar et al. 2017) designs a purpose-built hardware architecture for TNNs and implements it on FPGA and ASIC, which processes TNNs at up to $2.7\times$ better throughput, $3.1\times$ better energy efficiency and $635\times$ better area efficiency than the 1-bit implementation on TrueNorth chip. These works prove that TNN may be a better candidate to balance the trade off between efficiency and accuracy than BNN.

When training TNNs, however, all above works use simply thresholding operations to yield ternary weights. The differences among the methods just lie in the way of obtaining the threshold, which is either through a fixed threshold (Deng et al. 2018; Li, Zhang, and Liu 2016) or through a learned threshold optimized during training (Zhu et al. 2016). In this paper, we argue that such purely thresholding operations are not accurate enough to map the full-precision weights and easy to cause significant performance gap between TNNs with the full-precision counterparts. Thus we propose TRQ to introduce a ternary quantizer in a stem-

residual framework, which performs ternary quantization in a novelly recursive way and consequently improve the recognition performance.

Methodology

This section introduces our ternary neural networks with residual quantization (TRQ), which are designed to reduce the quantization error in a recursive way. In the following, we first give some preliminaries about ternarization. We then present our ternary quantization method, including the details on the forward and backward propagation. Furthermore, we describe how to generalize our approach to n-bit quantization and end by an extensive discussion.

Preliminary

The main operation in deep neural networks is expressed as

$$z = \mathbf{w}^\top \mathbf{a}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^n$ indicates the weight vector, $\mathbf{a} \in \mathbb{R}^n$ indicates the input activition vector computed by the previous network layer.

A ternary neural network means representing the floating-point weights and/or activitions with ternary values. Formally, the quantization can be expressed as

$$Q_x(\mathbf{x}) = \beta_x \mathbf{T}_\mathbf{x}, \tag{2}$$

where $\mathbf{x}$ indicates floating-point parameters including weights $\mathbf{w}$ and activitions $\mathbf{a}$, $\mathbf{T}_\mathbf{x}$ denotes ternary values after the quantization on $\mathbf{x}$. β_x is a scalar used to scale the ternary values, which can be computed from the floating-point parameters or learned via backpropagation. $\mathbf{T}_\mathbf{x}$ is usually obtained by thresholding function

$$\mathbf{T}_\mathbf{x} = \begin{cases} +1 & if\ \mathbf{x} > \Delta \\ 0 & if\ |\mathbf{x}| \leqslant \Delta, \\ -1 & if\ \mathbf{x} < -\Delta \end{cases} \tag{3}$$

where Δ denotes a fixed threshold used for quantization. With the ternary weights and activitions, the vector multiplications in the forward propagation can be reformulated as

$$z = Q_w(\mathbf{w})^\top Q_a(\mathbf{a}) = \beta_w \beta_a (\mathbf{T}_\mathbf{w} \odot \mathbf{T}_\mathbf{a}), \tag{4}$$

where $\odot$ represents the inner product for vectors with bit-wise operations.

In general, the derivative of quantization function $Q_x(\mathbf{x})$ is non-differentiable and thus unpractical to directly apply the backpropagation to perform the training phase. For this issue, we follow the now widely adopted "Straight-Through Estimator (STE)" (Hubara et al. 2016) to approximate the partial gradient calculation, which is formally expressed as

$$\frac{\partial Q_x(\mathbf{x})}{\partial \mathbf{x}} \approx \beta \mathbf{1}_{|\mathbf{x}| \leqslant 1}. \tag{5}$$

TNNs with Residual Quantization (TRQ)

Existing TNNs are based on directly thresholding method for ternary implementation, inevitably causing performance degradation due to an inaccurate mapping of full-precision values to ternary counterparts. To deal with the issue, Residual Quantization (TRQ) is introduced to learn TNNs. TRQ can extract binarized stem and residual respectively by performing recursive quantization on full-precision weights, which are combined to generate refined ternary representation, leading to the stem-residual framework for TNNs.

In our stem-residual ternarization framework, the stem is first extracted as a coarse fitting for full-precision weight $\mathbf{w}$, which is calculated by performing $\mathrm{sign}(\cdot)$ on $\mathbf{w}$ as

$$\mathbf{S}_\mathbf{w} = \alpha \mathrm{sign}(\mathbf{w}), \tag{6}$$

where α is a learnable coefficient, which avoids a very careful tuning to seek the optimal quantization scale compared with the previous methods. Then, we further calculate the quantization error as

$$\mathbf{R} = \mathbf{w} - \mathbf{S}_\mathbf{w} \tag{7}$$

Furthermore, we calculate the residual $\mathbf{R}_\mathbf{w}$ from $\mathbf{R}$ by performing $\mathrm{sign}(\cdot)$ on the quantization error $\mathbf{R}$

$$\mathbf{R}_\mathbf{w} = \alpha \mathrm{sign}(\mathbf{R}). \tag{8}$$

Based on Eq.6 and Eq.8, we finally obtain our ternary weight designed for more accurate approximation as

$$\mathbf{T}_\mathbf{w} = \mathbf{S}_\mathbf{w} + \mathbf{R}_\mathbf{w}. \tag{9}$$

Up to now, we achieve the ternary quantization in a stem-framework, with the full-precision weights quantitized to ternary values, *i.e.*, $\{-2\alpha, 0, 2\alpha\}$. Obviously, seeking a better coefficient α is significantly important for the effectiveness of quantizer, which would be elaborated in the following section.

Backward Propogation of TRQ

In the backward propagation, what need to be learned and updated are the full-precision weight **w** and the learnable coefficient α. For the stem-residual framework, the two kinds of parameters are jointly learned. And in each layer, TRQ updates the **w** first and then the α.

Update w For **w** updating, the gradient through the quantizer to weights are estimated by a STE that pass the gradient whose weight value is in the range of (-2α, 2α):

$$\frac{\partial \mathbf{T}_\mathbf{w}}{\partial \mathbf{w}} = \mathbf{1}_{|\mathbf{x}| \leqslant 2\alpha}. \tag{10}$$

Then, we can obtain the updating process of $\mathbf{w}$

$$\delta_\mathbf{w} = \frac{\partial L}{\partial \mathbf{T}_\mathbf{w}} \frac{\partial \mathbf{T}_\mathbf{w}}{\partial \mathbf{w}}, \tag{11}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \delta_\mathbf{w}, \tag{12}$$

where L is the loss function and η is learning rate.

Update α The coefficient α determines the scale of binarized stem and residual, which is directly related to the quality of the ternary weights. Moreover, we also empirically find the recognition performance is quite sensitive to the α. Thus rather than a coarse gradient acquired like **w**, we disassemble the quantizer to calculate a finer gradient of α

$$\frac{\partial \mathbf{T_w}}{\partial \alpha} = \text{sign}(\mathbf{w}) + \text{sign}(\mathbf{R}) + \alpha \frac{\partial \text{sign}(\mathbf{R})}{\partial \alpha} \tag{13}$$

where

$$\begin{aligned} \frac{\partial \text{sign}(\mathbf{R})}{\partial \alpha} &= \frac{\partial \text{sign}(\mathbf{R})}{\partial \mathbf{R}} \frac{\partial \mathbf{R}}{\partial \alpha} \\ &= \mathbf{1}_{|\mathbf{R}| \leqslant 1} \left(-\text{sign}(\mathbf{w})\right). \end{aligned} \tag{14}$$

Then, we can obtain the updating process of α

$$\delta_\alpha = \sum \frac{\partial L}{\partial \mathbf{T_w}} \frac{\partial \mathbf{T_w}}{\partial \alpha}, \tag{15}$$

$$\alpha \leftarrow \alpha - \eta \delta_\alpha. \tag{16}$$

Generalization to n-bit Quantization

We focus on ternary quantization in this paper, while it does not mean that TRQ is limited to ternary applications. Actually, TRQ could also be generalized to multiple bits by recursively encoding residual. In this section we propose a feasible scheme for TRQ expansion, which is not the only way and could be further explored in the future work.

We obtain the subtly quantized weights by recursively performing quantization on full-precision weights. In this process, residual at different quantization levels is generated for refining the quantitized weights. Here for n-bit ($n = 2, 3, 4, ...$) quantization, we define the residual at level i ($i = 1, 2, ..., 2^n - 3$) as $\mathbf{R}^i_\mathbf{w}$, which could be computed as

$$\mathbf{R}^i_\mathbf{w} = \alpha \text{sign}(\mathbf{w} - \mathbf{T}^{i-1}_\mathbf{w}), \tag{17}$$

where $\mathbf{T}^{i-1}_\mathbf{w}$ denotes the quantized weights at $(i-1)$th level, and we recursively acquire the quantized weights at level i as

$$\mathbf{T}^i_\mathbf{w} = \mathbf{T}^{i-1}_\mathbf{w} + \mathbf{R}^i_\mathbf{w}. \tag{18}$$

Here we regard the ternary quantization as the initial state for recursive quantization as

$$\mathbf{T}^0_\mathbf{w} = \mathbf{S}_\mathbf{w} + \alpha \text{sign}(\mathbf{w} - \mathbf{S}_\mathbf{w}). \tag{19}$$

Based on such recursive quantization, we could easily obtain the residual at different levels, thus refining the residual and reducing the approximation error with the full-precision counterparts.

For the updating of α in backward propagation, due to the complexity of recursive process, we just roughly estimate the gradient α by regarding it as the coefficient of $\mathbf{S_w}$ and $\mathbf{R}^i_\mathbf{w}$

$$\frac{\partial \mathbf{T_w}}{\partial \alpha} = \mathbf{S_w} + \sum_{i=0}^{2^n-3} \mathbf{R}^i_\mathbf{w}. \tag{20}$$

Complexity Analysis

A comprehensive comparison on computational complexity is shown in Table 1. We assume that the input number of the neuron is N, *i.e.*, N inputs and one neuron output. For computational complexity of TNNs, we follow the setting of GXNOR-Net (Deng et al. 2018) for a comparison. As described in GXNOR-Net, with the event-driven paradigm, the resting computation would occur when the weight or activation of TNNs is zero, and the exception cases are achieved by XNOR operations. As a result, the computational complexity of TNNs is similar to BNNs, half of the network with 1-bit weights and 2-bit activations. Noted that for TRQ, the stem-residual framework is only employed on weights, thus it also enjoys the low complexity $O(N)$ as normal TNNs.

Differences of TRQ from Existing Residual Quantization Methods

Residual quantization has been first proposed in High-Order Residual Quantization (HORQ) (Li et al. 2017) to enhance the performance of BNNs, further being explored by (Guo et al. 2017; Fromm, Patel, and Philipose 2018) to be encoded into low-bitwidth CNNs. All above works compute residual error and recursively approximate it by a series of binary maps. However, limited by the residual scales, they can be just applied to n-bit quantization, with no generalization ability to arbitrary value quantization even parameter changes, such as the TNNs emphasized in this paper. Instead, our TRQ enjoys the flexibility by the skillful combination of the binarized stem and residual, thus enabling ternary quantization and even arbitrary value quantization by recursively refining the residual. Moreover, a key feature of the prior residual schemes is the use of analytically calculated scaling coefficients, which can be sub-optimal. In contrast, our TRQ employs learnable coefficient α to minimize the training loss, thus fully utilizing the strength of back propagation algorithm to seek for the suitable quantization scale automatically.

Experiments

In this section, to demonstrate the effectiveness of the proposed TRQ, we perform diverse experiments on three classification datasets: CIFAR-10/100 (Alex Krizhevsky 2014) and ImageNet (ILSVRC 2012) (Russakovsky et al. 2015). Notice that in the following sections, unless otherwise specified, we refer the **baseline** as the quantization network with same architecture as TRQ, while adopting a normal ternary method described in Section 3.1 (Preliminary).

Implementation Details

Data preprocessing. For CIFAR-10/100, all the images are padded with 4 pixels on each side, then a random 32×32 crop is applied, followed by a random horizontal flip. During inference, the scaled images are used without any augmentation. For ImageNet, training images are randomly cropped into the resolution of 224×224. After that, the images are normalized using the mean and standard deviation. No additional augmentations are performed except the random horizontal flip. However, for validation images, we use center crop instead of random crop and no flip is applied.

Training procedure. We conduct experiments mainly on ResNet (He et al. 2016) backbones, including ResNet-18 and ResNet-34. VGG-Small (Simonyan and Zisserman 2014) is also leveraged for the CIFAR-10 and CIFAR-100 experiments. Similar with previous works (Liu et al. 2018; Kim et al. 2020; Gu et al. 2019a), we do not quantize the first and last layers. For experiments on CIFAR-10/100, we run

Bit width(A/W)	Operations				Complexity
	Multiplication	Accumulation	XNOR	BitCount	
32/32	N	N	0	0	-
1/1	0	0	N	1	O(N)
2/1	0	0	2N	1	O(2N)
ter/ter	0	0	0∼N	0/1	O(N)

Table 1: Operation overhead comparisons with different computing bit width.

	Width	TRQ-wo	TRQ	TRQ-0.6
ResNet-18	16-16-32-64	52.1	54.9	53.9
ResNet-18	32-32-64-128	60.5	62.7	61.3
VGG-Small	-	62.6	65.4	60.5

Table 2: The accuracy (%) of TRQ with and without α (TRQ and TRQ-wo), and with fixed $\alpha = 0.6$ (TRQ-0.6) on CIFAR-100.

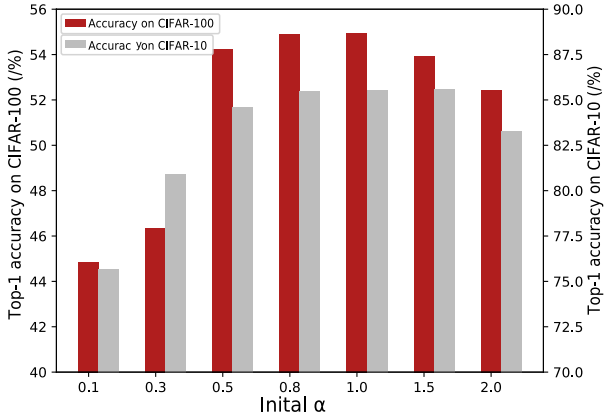


Figure 2: The Top-1 accuracy (%) on CIFAR-10 and CIFAR-100 with different initial α.

the training algorithm for 200 epochs with a batch size of 256. Besides, a linear learning rate decay scheduler is used, and the initial learning rate is set to 0.01. For experiments on ImageNet, we train the models for up to 100 epochs with a batch size of 256. The learning rate starts from 0.001 and is decayed twice by multiplying 0.1 at 75th and 95th epoch. For all settings, Adam with momentum of 0.9 is adopted as the optimizer.

Ablation Study on CIFAR

In this section, we first perform hyperparameter sweeps to determine the value of initial α to use. Following this we analyze the necessary of α, then show TRQ's generalization to multiple bits, and finally we evaluate the effectiveness of TRQ on CIFAR datasets.

Initial value of α. The initiation of parameters is always important for network training. Thus we set different initial values 0.10, 0.3, 0.5, 0.8, 1, 1.5, and 2 to α, to explore their influence on classification. The experiments are performed on the CIFAR-10/100 with ResNet-18 backbone. From the results on CIFAR-10 in Figure 3, we can observe that the performance is similar when the initial values of α are set between 0.8 and 1.5, and the best performance can be obtained at 1.5. Meanwhile, from the results on CIFAR-100, the good performance plateau appears when initial α is at the range between 0.8 and 1, and the performance of initial value 1 performs slightly better than that of 0.8. For both CIFAR-10 and CIFAR-100, the performance of initial values outside the 0.5 to 1.5 is fairly worse, which shows the importance of setting the initial value of α carefully. Based on the above discoveries, we set the initial value of α as 1 in the following experiments, which shows a stably high classification performance on both two datasets.

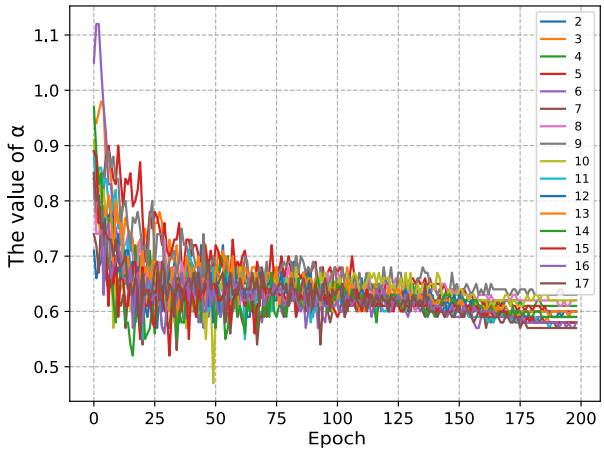


Figure 3: Evolution of α values in different layers during training with ResNet-18 backbone on CIFAR-100.

Analysis of α. α is introduced in stem-residual framework to automatically seek for a reasonable quantization scale. To valid the necessarity of α, we provide the experiments with and without α on CIFAR-100 with the backbone ResNet-18. As shown in Table 2, compared with the TRQ without α (TRQ-wo), TRQ achieves better performance by a large margin (more than 2%), thus indicating that α is quite important for training TRQ.

Simultaneously, as illustrated in Figure 3, we explore how the value of α changes during training. It can be observed that α converges to around 0.6 with training. While this doesn't mean that α should be fixed and not optimized. As shown in Table 2, we compare the results in two cases, *i.e.*, α is fixed to 0.6 (TRQ-0.6) and α is optimized by back-propagation. As we can see, when fixed α as 0.6, a greater performance decrease happens. When employed with VGG-Small backbone, the accuracy even drops nearly 5% compared with the learnable α, thus validating the superiority of the leanable α. We conjecture that is because with the learn-

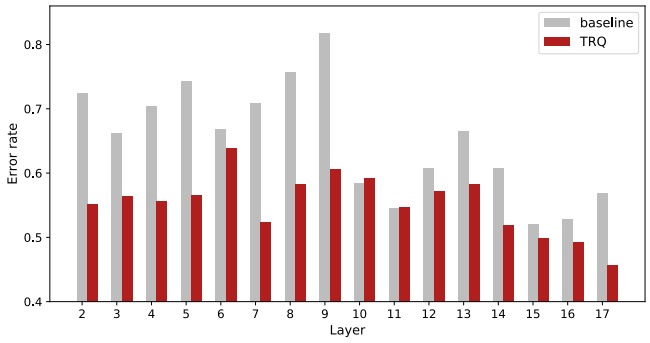


Figure 4: Quantization error of TRQ and baseline based on ResNet-18 backbone.

		CIFAR-10/%	CIFAR-100/%
ResNet-18 16-16-32-64	full-precision	87.7	58.2
	baseline	85.2	54.8
	TRQ	85.5	54.9
ResNet-18 32-32-64-128	full-precision	90.9	63.0
	baseline	87.5	60.6
	TRQ	89.3	62.7
VGG-Small	full-precision	92.6	66.8
	baseline	89.1	61.8
	TRQ	91.2	65.4

Table 3: The experimental comparison of baseline and TRQ on CIFAR datasets.

able α in stem-residual framework, the quantizer could be automatically finetuned to find the best quantization mapping for each layer, thus yielding better performance than the fixed case.

Quantization Error. In order to better understand our TRQ, which achieves more accurate mapping between ternary weights and their full-precision counterparts, we adopt mean square error (MSE) (Esser et al. 2019) to calculate the quantization error between $\mathbf{w}$ and $\mathbf{T_w}$

$$\mathbf{E} = \frac{1}{M}\sum(\frac{\mathbf{w}-\mathbf{T_w}}{\mathbf{w}})^2, \tag{21}$$

where M denotes the total number of weights in each layer. In Figure 4, we plot the quantization error for the 2th-17th layer of ResNet-18. The results show our methods (the red histogram) have lower quantization error compared with baseline (the gray histogram) which achieved with the method in Section 3.1 in most layers. In particular, the quantization error can be reduced by more than 25% (0.8 *vs* 0.6) in the 9th layer.

Generalization to n-bit Quantization. We illustrate that our TRQ can not only improve the performance on ternary quantization, but also could be generalized to multiple bits. Here we adopt the expansion method described in Sec. 3.4, and perform the experiments on CIFAR-100 with the backbone of ResNet-18. The baseline model is implemented as the way as DoReFa-Net (Zhou et al. 2016). As shown in Figure 5, we can see that the accuracy of TRQ increases

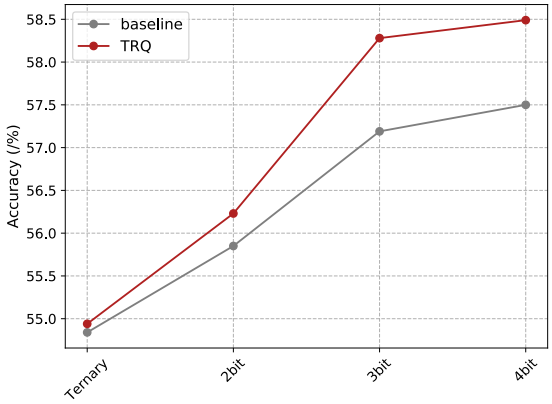


Figure 5: The results of TRQ with multi-bits expansion on CIFAR-100.

($56.2\% \rightarrow 58.3\% \rightarrow 58.5\%$) as the bit width increases from 2bit to 4bit, indicating that the compound residual at multi-levels could refine the quantized weights thus improving the recognition accuracy. Moreover, our TRQ consistently surpasses the baseline on each bit width ($0.4\%, 1.1\%, 1.0\%$ on 2bit, 3bit and 4bit, respectively), which demonstrates the superiority and potential of the residual quantization on multiple bits.

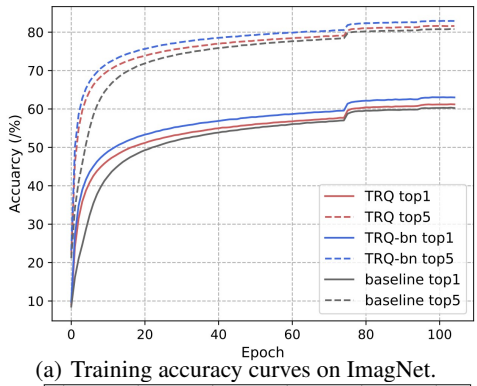
Evaluation on CIFAR. To valid the effectiveness of TRQ, here we perform ablation evaluation on CIFAR datasets. Three backbones are used in this experiment, including VGG-Small, ResNet-18 with the width of 16-16-32-64 and 32-32-64-128. We report the performance of baseline and TRQ on both CIFAR-10 and CIFAR-100 in Table 3. As shown in Table 3, for ResNet-18, TRQ achieves stable improvement on both CIFAR-10 and CIFAR-100 datasets compared with the corresponding baseline. Moreover, TRQ with the backbone ResNet-18 whose width is 32-32-64-128 even realizes nearly lossless ternarization on CIFAR-100 (only with a 0.3% performance drop). All these demonstrate the effectiveness of TRQ on ResNet. For VGG-Small, our TRQ consistently surpasses the baseline by a margin of 2.1% and 3.5% on CIFAR-10 and CIFAR-100, respectively, which further shows the general improvement brought by TRQ.

Comparison on ImageNet

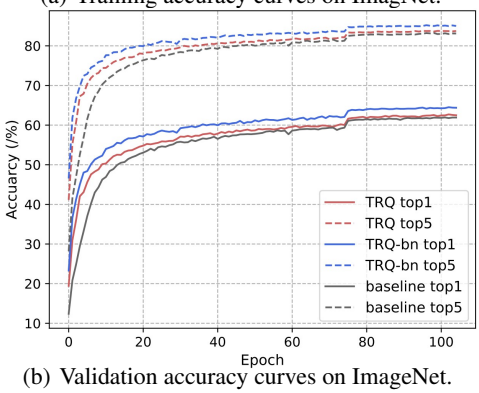
We further analyze the effectiveness of TRQ on the large-scale dataset ImageNet. Since the dataset is challenge for network optimization, we use multi-batchnormalization (multi-bn) strategy on ResNet architecture to alleviate optimization problems, which termed as TRQ-bn in the experiment. For a basic block in TRQ-bn, three batch normalizaton layers are employed: the first is a pre-bn (Zhang et al. 2018) before quantization, the second is a normal bn following the ternary convolutional layer, and the last is an additional bn following the shortcut. Such multi-bn can significantly improve the network performance by improving the distribution of feature maps with only small additional memory and computation.

Network	Method	A/W	Top-1/%	Top-5/%	Complexity
ResNet-18	full-precision	32/32	69.3	89.2	-
	baseline	ter/ter	61.6	82.7	O(N)
	TRQ(ours)	ter/ter	62.6	83.7	O(N)
	TRQ-bn(ours)	ter/ter	64.4	85.1	O(N)
	TRQ-a(ours)	ter/ter	65.7	85.9	O(N)
	RTN	ter/ter	64.5	-	O(N)
	XNOR-Net	1/1	51.2	73.2	O(N)
	BiReal-Net	1/1	56.4	79.5	O(N)
	LQ-Net	2/1	62.6	84.3	O(2N)
ResNet-34	full-precision	32/32	73.3	91.3	-
	baseline	ter/ter	65.2	85.7	O(N)
	TRQ(ours)	ter/ter	66.2	86.3	O(N)
	TRQ-bn(ours)	ter/ter	68.2	87.7	O(N)
	BiReal-Net	1/1	62.2	83.9	O(N)
	LQ-Net	2/1	66.6	86.9	O(2N)
	HWGQ	2/1	64.3	85.7	O(2N)

Table 4: Comparison of top-1 and top-5 accuracy on ImagNet.



(a) Training accuracy curves on ImagNet.



(b) Validation accuracy curves on ImageNet.

Figure 6: Accuracy curves of baseline, TRQ and TRQ-bn with ResNet-18 backbone on ImageNet.

We illustrate the training and validation accuracy curves of baseline, TRQ and TRQ-bn in Figure 6, which are based on a ResNet-18 backbone. From Figure 6, we can observe that TRQ greatly improves the convergence speed of TNNs. Simultaneously from the results in Table 4, TRQ improves baseline by 1.0% on both ResNet-18 and ResNet-34 top-1 accuracy, which validates the effectiveness of our TRQ on large-scale dataset. Moreover, TRQ-bn could further obtains an improvement of about 2% on both the two networks, which finally achieves approximate 93% of the accuracy of their full-precision counterparts.

To evaluate the overall performance of TRQ, we further compare TRQ with four state-of-the-art quantization on ImageNet, *i.e.*, XNOR-Net (Rastegari et al. 2016), BiReal-Net (Liu et al. 2018), LQ-Net (Zhang et al. 2018), HWGQ (Cai et al. 2017) and RTN (Li et al. 2020). To perform fair comparison with RTN whose quantization procedure of weight and activation are both improved, we apply residual quantization to activation as well, leading to TRQ-a. The results are reported in Table 4. From Table 4, by comparing with the state-of-the-art BNNs including XNOR-Net and BiReal-Net, we can significantly boost the performance. For example, TRQ outperforms XNOR-Net and BiReal-Net by 11% and 6% on ResNet-18, respectively. It is because that ternary values $\{-1, 0, 1\}$ has stronger representational capability than binary values $\{-1, 1\}$, while the complexity of two methods are same because of the event-driven paradigm in TNNs. Moreover, our TRQ can even achieve better performance than the methods with $O(2N)$ complexity, including the "A/W = 2/1" cases in LQ-Net and HWGQ. Besides, our TRQ-a surpasses RTN 1.2% in accuracy, demonstrating the advantage of the effective residual quantization scheme.

Conclusion

In this paper, we propose efficient and accurate ternary neural networks equipped with residual quantization (TRQ). Instead of prior works that directly apply thresholding quantization, our TRQ implement ternary quantization from a stem-residual perspective. Particularly, TRQ rethinks of ternary weights as a combination of binarized stem and residual, thus endowing the ternary quantizer more accurate mapping between full-precison weights and ternary counterparts. Furthermore, we empirically demonstrate that TRQ is generic to extend to multiple bits through recursively encoded residual, which ulteriorly brings improvement on recognition accuracy. As a result, TRQ can significantly boot the performance of BNNs, and additionally even outperforms quantization methods with higher complexity.

Acknowledgments

This work is supported by National Natural Science Foundation of China (U20B2042), National Natural Science Foundation of China (62076019) and Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence" under Grant 2020AAA0108201. Wenrui Ding is the corresponding author.

References

Alemdar, H.; Leroy, V.; Prost-Boucle, A.; and Pétrot, F. 2017. Ternary neural networks for resource-efficient AI applications. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2547–2554. IEEE.

Alex Krizhevsky, Vinod Nair, G. H. 2014. The cifar-10 dataset. *online:http://www.cs.toronto.edu/kriz/cifar.html* 4.

Bulat, A.; and Tzimiropoulos, G. 2019. XNOR-Net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863* .

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5918–5926.

Deng, L.; Jiao, P.; Pei, J.; Wu, Z.; and Li, G. 2018. GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks* 100: 49–58.

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

Fromm, J.; Patel, S.; and Philipose, M. 2018. Heterogeneous bitwidth binarization in convolutional neural networks. In *Advances in Neural Information Processing Systems*, 4006–4015.

Gu, J.; Li, C.; Zhang, B.; Han, J.; Cao, X.; Liu, J.; and Doermann, D. 2019a. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8344–8351.

Gu, J.; Zhao, J.; Jiang, X.; Zhang, B.; Liu, J.; Guo, G.; and Ji, R. 2019b. Bayesian Optimized 1-Bit CNNs. In *Proceedings of the IEEE International Conference on Computer Vision*, 4909–4917.

Guo, Y.; Yao, A.; Zhao, H.; and Chen, Y. 2017. Network sketching: Exploiting binary structure in deep cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5955–5963.

Han, K.; Wang, Y.; Xu, Y.; Xu, C.; Wu, E.; and Xu, C. 2020. Training binary neural networks through learning with noisy supervision. In *International Conference on Machine Learning*, 4017–4026. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hou, L.; Yao, Q.; and Kwok, J. T. 2016. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600* .

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *Advances in neural information processing systems*, 4107–4115.

Jung, S.; Son, C.; Lee, S.; Son, J.; Han, J.-J.; Kwak, Y.; Hwang, S. J.; and Choi, C. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4350–4359.

Kim, H.; Kim, K.; Kim, J.; and Kim, J.-J. 2020. BinaryDuo: Reducing Gradient Mismatch in Binary Activation Network by Coupling Binary Activations. *arXiv preprint arXiv:2002.06517* .

Laborieux, A.; Bocquet, M.; Hirtzlin, T.; Klein, J.-O.; Diez, L. H.; Nowak, E.; Vianello, E.; Portal, J.-M.; and Querlioz, D. 2020. Low Power In-Memory Implementation of Ternary Neural Networks with Resistive RAM-Based Synapse. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 136–140. IEEE.

Li, F.; Zhang, B.; and Liu, B. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711* .

Li, Y.; Dong, X.; Zhang, S. Q.; Bai, H.; Chen, Y.; and Wang, W. 2020. RTN: Reparameterized Ternary Network. In *AAAI*, 4780–4787.

Li, Z.; Ni, B.; Zhang, W.; Yang, X.; and Gao, W. 2017. Performance guaranteed network acceleration via high-order residual quantization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2584–2592.

Liu, Z.; Luo, W.; Wu, B.; Yang, X.; Liu, W.; and Cheng, K.-T. 2020a. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision* 128(1): 202–219.

Liu, Z.; Shen, Z.; Savvides, M.; and Cheng, K.-T. 2020b. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. *arXiv preprint arXiv:2003.03488* .

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K.-T. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, 722–737.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 525–542. Springer.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115(3): 211–252.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Yang, Z.; Wang, Y.; Han, K.; Xu, C.; Xu, C.; Tao, D.; and Xu, C. 2020. Searching for low-bit weights in quantized neural networks. *Advances in Neural Information Processing Systems* 33.

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

Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* .

Zhu, C.; Han, S.; Mao, H.; and Dally, W. J. 2016. Trained ternary quantization. *arXiv preprint arXiv:1612.01064* .

Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; and Reid, I. 2019. Structured binary neural networks for accurate image classification and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 413–422.