2019 IEEE Winter Conference on Applications of Computer Vision

Optimize Deep Convolutional Neural Network with Ternarized Weights and High Accuracy

Zhezhi He
Department of ECE
University of Central Florida
Elliot.he@knights.ucf.edu

Boqing Gong
Tencent AI Lab
Bellevue, WA 98004
boqinggo@outlook.com

Deliang Fan
Department of ECE
University of Central Florida
dfan@ucf.edu

Abstract

Deep convolution neural network has achieved great success in many artificial intelligence applications. However, its enormous model size and massive computation cost have become the main obstacle for deployment of such powerful algorithm in the low power and resource limited embedded systems. As the countermeasure to this problem, in this work, we propose statistical weight scaling and residual expansion methods to reduce the bit-width of the whole network weight parameters to ternary values (i.e. -1, 0, +1), with the objectives to greatly reduce model size, computation cost and accuracy degradation caused by the model compression. With about 16× model compression rate, our ternarized ResNet-32/44/56 could outperforms full-precision counterparts by 0.12%, 0.24% and 0.18% on CIFAR-10 dataset. We also test our ternarization method with AlexNet and ResNet-18 on ImageNet dataset, which both achieve the best top-1 accuracy compared to recent similar works, with the same 16× compression rate. If further incorporating our residual expansion method, compared to the full-precision counterpart, our ternarized ResNet-18 even improves the ***top-5 accuracy by 0.61%*** *and merely degrades the top-1 accuracy only by 0.42% for ImageNet dataset, with 8× model compression rate. It outperforms the recent ABC-Net by 1.03% in top-1 accuracy and 1.78% in top-5 accuracy, with around 1.25× higher compression rate and more than 6× computation reduction due to the weight sparsity.*

1. Introduction

Deep convolutional neural networks (CNNs) have taken an important role in artificial intelligence algorithm which has been widely used in computer vision, speech recognition, data analysis and etc [9]. Nowadays, deep CNNs become more and more complex consisting of more layers, larger model size and denser connections. However, from the hardware acceleration point of view, deep CNNs still suffer from the obstacle of hardware deployment due to their massive cost in both computation and storage. For instance, VGG-16 [15] from ILSVRC 2014 requires 552MB of parameters and 30.8 GFLOP per image, which is difficult to deploy in resource limited mobile systems. Research has shown that deep CNN contains significant redundancy, and the state-of-the-art accuracy can also be achieved through model compression [6]. Many recent works have been proposed to address such high computational complexity and storage capacity issues of existing deep CNN structure using model compression techniques.

As one of the most popular model compression technique, weight quantization techniques are widely explored in many related works [6,13,16–18] which can significantly shrink the model size and reduce the computation complexity. It is worthy to note that weight binarization (-1, +1) or ternarization (-1, 0, +1) is more preferred compared to other methods since floating point multiplication is not needed and most complex convolution operations are converted to bit-wise *xnor* and *bit-count* operations, which could greatly save power, area and latency. Meanwhile, it does not modify the network topology or bring extra computation to the system, which may increase hardware deployment complexity, such as pruning or hash compression. However, for compact deep CNN models that are more favored by hardware deployment, the existing aggressive weight binarization or ternarization methods still encounter large accuracy degradation for large scale benchmarks.

In this work, we propose statistical weight scaling and residual expansion methods to convert the entire network weight parameters to ternary format (i.e. -1, 0, +1), with objectives to greatly reduce model size, computation cost and accuracy degradation caused by model compression. Our main contributions in this work can be summarized as:

- A iterative statistical weight ternarization method is used to optimize the layer-wise scaling factor to mitigate accuracy degradation caused by aggressive weight

978-1-7281-1975-5/19/$31.00 ©2019 IEEE
DOI 10.1109/WACV.2019.00102

ternarization. Such method leads to that the ternarization of ResNet32/44/56 even outperforms full-precision counterparts by 0.12%, 0.24% and 0.18% for CIFAR-10 dataset. Moreover, both the ternarization of AlexNet and ResNet-18 in ImageNet dataset achieve the best top-1 accuracy compared to recent similar works, with the same 16X compression rate.

- Different from other existing works that leave the network's first and last layers in full-precision, we ternarize all the convolution and fully-connected layers, where a very limited accuracy degradation is observed. Such whole network ternarization could bring large amount of power, area and latency saving for domain-specific deep CNN accelerators.

- To further mitigate accuracy degradation from the full-precision baseline, we introduce a residual expansion methodology. Compared to full-precision network, our ternarized ResNet-18 even improves top-5 accuracy by 0.61% and degrades top-1 accuracy only by 0.42% for ImageNet dataset, with 8× model compression rate. It outperforms the recent ABC-Net by 1.03% in top-1 accuracy and 1.78% in top-5 accuracy, with 1.25× higher compression rate.

2. related works

Recently, model compression on deep convolutional neural network has emerged as one hot topic in the hardware deployment of artificial intelligence. As the most popular technique, weight quantization techniques are widely explored in many related works which can significantly shrink the model size and reduce the computation complexity. Among all those works, DCNN with binary weight is the most discussed scheme, since it leads to 32x model compression rate. More importantly, it also converts the original floating-point multiplication (i.e. *mul*) operations into addition/subtraction (i.e. *add/sub*), which could greatly reduce computational hardware resources and further dramatically lowers the existing barrier to deploy powerful deep neural network algorithm into low power resource-limited embedded system. BinaryConnect [4] is the first work of binary CNN which can get close to the state-of-the-art accuracy on CIFAR-10, whose most effective technique is to introduce the gradient clipping. After that, both BWN in [13] and DoreFa-Net [17] show better or close validation accuracy on ImageNet dataset. In order to reduce the computation complexity to the bone, XNOR-Net [13] binarize the input tensor of the convolution layer which further converts the *add/sub* operations into bit-wise *xnor* and *bit-count* operations. Besides weight binarization, there are also recent works proposing to ternarize the weights of neural network using trained scaling factors [18]. Leng et. al. employ ADMM method to optimize neural network weights in configurable discrete levels to trade off between accuracy and model size [10]. ABC-Net in [12] proposes multiple parallel binary convolution layers to improve the network model capacity and accuracy, while maintaining binary kernel. All above discussed aggressive neural network binarization or ternarization methodologies sacrifice the inference accuracy in comparison with the full precision counterpart to achieve large model compression rate and computation cost reduction.

3. Weight Ternarization Training Method

3.1. Ternarization with iterative statistical scaling

As shown in Fig. 1, our training methodology to obtain accurate deep Convolutional Neural Network (CNN) [1] with ternarized weights can be divided into two main steps: Initialization and iterative weight ternarization training. We first train a designated neural network model with full precision weights to act as the initial model for the future ternarized neural network model. After model initialization, we retrain the ternarized model with iterative inference (including *statistical weight scaling*, weight ternarization and ternary weight based inference for computing loss function) and back-propagation to update full precision weights.

Initialization: The reason that we use a pretrained model with full precision weights as the initial model comes in twofold: 1) Fine-tuning from the pretrained model normally gives higher accuracy for the quantized neural network. 2) Our ternarized model with the initialized weights from pretrained model converges faster in comparison to ternarization training from scratch. For practical application, engineers will definitely use a full-precision model to estimate the ideal upper-bound accuracy. Afterwards, various model compression techniques are applied to the full-precision model to compress model size while trying to mitigate accuracy loss as well.

Iterative weight ternarization training: After loading the full-precision weights as initialization state, we start to fine-tune the ternarized CNN model. As described in Fig. 1, there are three iterative steps for the following training, namely, ①-statistical weight scaling and weight ternarization, ②-ternary weight based inference for loss function computation and ③-back propagation to update full precision weights.

① will first ternarize current full precision weights and compute the corresponding scaling factor based on current statistical distribution of full precision weight. For weight ternarization (i.e. -1, 0, +1), we adopt the variant staircase ternarization function, which compares the full precision weight with the symmetric threshold $\pm\Delta_{th}$ [11]. Such

[1] In this work, all the convolution kernels and fully-connected layers do not have bias term, excluding the last layer.

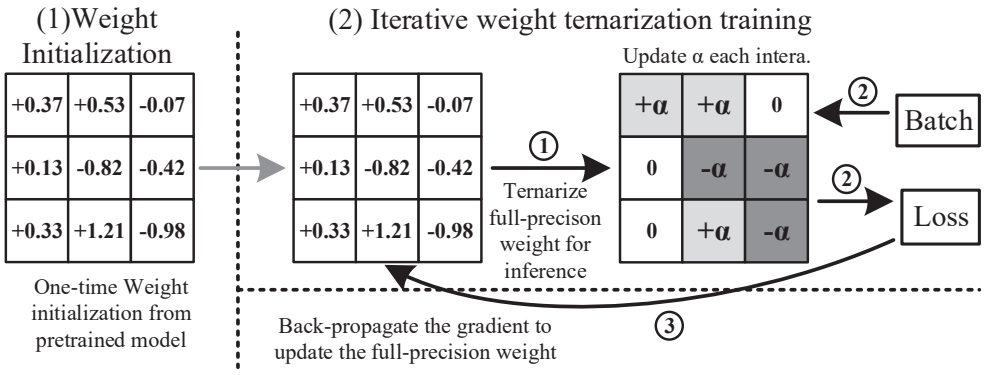


Figure 1. The training scheme of ternarized CNN model. ①-③ are iteratively operated during the training.

weight ternarization function in the forward path for inference can be mathematically described as:

$$\text{Forward}: \quad \boldsymbol{w}_l' = \begin{cases} \alpha \times Sign(w_{l,i}) & |w_{l,i}| \geq \Delta_{th} \\ 0 & |w_{l,i}| < \Delta_{th} \end{cases} \quad (1)$$

where $\boldsymbol{w}_l$ denotes the full precision weight tensor of layer l, $\boldsymbol{w}_l'$ is the weight after ternarization. We set the scaling factor as:

$$\alpha = E(|\boldsymbol{w}_{l,i}|), \quad \forall\{i \big| |\boldsymbol{w}_{l,i}| \geq \Delta_{th}\} \quad (2)$$

which statistically calculates the mean of absolute value of designated layer's full precision weights that are greater than the threshold Δ_{th}. Unlike TWN uses $\Delta_{th} = 0.7 \times E(\boldsymbol{w}_l)$ as threshold, we set $\Delta_{th} = \beta \times max(|\boldsymbol{w}_l|)$ as threshold [11, 18]. The reason is that, for each training iteration, the weight update causes large value drifting of $E(\boldsymbol{w}_l)$, which correspondingly leads to unstable Δ_{th}. In this work, we employ single scaling factor for both positive and negative weights, since such symmetric weight scaling factor can be easily extracted and integrated into following Batch Normalization layer or ReLU activation function (i.e., both perform element-wise function on input tensor) for the hardware implementation.

Then, in ②, the input mini-batch takes the ternarized model for inference and calculates loss w.r.t targets. In this step, since all of the weights are in ternary values, all the dot-product operations in layer l can be expressed as:

$$\boldsymbol{x}_l^T \cdot \boldsymbol{w}_l' = \boldsymbol{x}_l^T \cdot (\alpha \cdot Tern(\boldsymbol{w}_l)) = \alpha \cdot (\boldsymbol{x}_l^T \cdot Tern(\boldsymbol{w}_l)) \quad (3)$$

where $\boldsymbol{x}_l$ is the vectorized input of layer l. Since $Tern(\boldsymbol{w}_{l,i}) \in \{-1, 0, +1\}$, $\boldsymbol{x}_l^T \cdot Tern(\boldsymbol{w}_l)$ can be easily realized through addition/subtraction without multi-bit or floating point multiplier in the hardware, which greatly reduces the computational complexity.

In ③, the full precision weights will be updated during back-propagation. Then, the next iteration begins to re-compute weight scaling factor and ternarize weights as described in step-①. Meanwhile, since the ternarization function owns zero derivatives almost everywhere, which makes it impossible to calculate the gradient using chain rule in backward path. Thus, the Straight-Through Estimator (STE) [1] [17] of such ternarization function in the back-propagation is applied to calculate the gradient as follow:

$$\text{Backward}: \quad \frac{\partial g}{\partial w} = \begin{cases} \frac{\partial g}{\partial w'} & if\ |w| \leq 1 \\ 0 & otherwise \end{cases} \quad (4)$$

where the gradient clipping prevents the full precision weight growing too large and stabilize the threshold $\beta \times max(|\boldsymbol{w}_l|)$ during training (i.e., fine-tuning), thus leading to higher inference accuracy ultimately.

3.2. Residual expansion to improve accuracy

As the aforementioned discussion in the introduction, works like DoreFa-net [6, 17] adopt the multilevel quantization scheme to preserve the model capacity and avoid the accuracy degradation caused by extremely aggressive (i.e. binary/ternary) weight quantization. Since our main objective is to totally remove floating point multiplication and only use addition/subtraction to implement the whole deep CNN, while minimizing accuracy degradation caused by weight ternarization. We propose a *Residual Expanded Layer* (REL) as the remediation method to reduce accuracy degradation. As depicted in Fig. 2, we append one REL over the convolution layers and fully-connected layers in the original network structure, where the expansion factor T_{ex} is 2 in this case. We use *Conv* and *Conv$_r$* to denote the original layer and the added REL respectively. Both *Conv* and *Conv$_r$* are ternarized from the identical full precision parameters. During training, the network structure with RELs follow the same procedures as discussed in Fig. 1. The only difference is that *Conv* and *Conv$_r$* has two different threshold factors (e.g., $\beta = \{0.05, 0.1\}$ are in this work for $T_{ex} = 2$). We could append more RELs for compact CNN with low network redundancy.

The appended REL equivalently introduces more num-

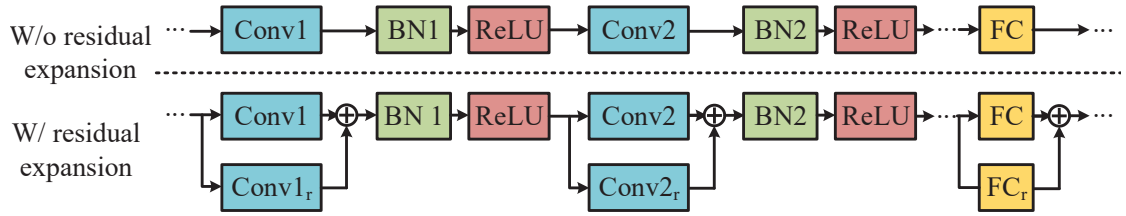


Figure 2. The block diagram of (top) original network topology and (bottom) the network with residual expansion to further compensate the accuracy degradation. In this figure, the expansion factor T_{ex} is 2.

ber of levels to each kernel. Assuming two threshold factors a, b for *Conv* and *Conv*$_r$ layers, where $a > b$, then we can formulate their computation as:

$$\boldsymbol{x}^T \cdot \boldsymbol{w}' + \boldsymbol{x}^T \cdot \boldsymbol{w}'_r = \boldsymbol{x}^T \cdot (\alpha \cdot \underset{\beta=a}{Tern}(\boldsymbol{w}) + \alpha_r \cdot \underset{\beta=b}{Tern}(\boldsymbol{w}_r)) \quad (5)$$

where the function $\alpha \cdot \underset{\beta=a}{Tern}(\boldsymbol{w}) + \alpha_r \cdot \underset{\beta=b}{Tern}(\boldsymbol{w}_r)$ is equivalent to a multi-threshold function that divides the element in $\boldsymbol{w}$ into levels of $\gamma = \{-\alpha - \alpha_r, -\alpha, 0, \alpha, \alpha + \alpha_r\}$, with thresholds $\{-b, -a, a, b\}$. Enlarging the expansion factor T_{ex} will equivalently introduce more number of levels for the weights and recover the ternarized model capacity towards its full-precision baseline. However, compared to the case that directly use a multi-bit kernels, the actual convolution kernel is still in ternary format and thus no multi-bit multiplication is needed (see more discussions in Section 5.3).

4. Experiments

In this work, all experiments are performed under the framework of Pytorch. The performance of our neural network ternarization methodology is examined using two datasets: CIFAR-10 and ImageNet. Other small datasets like MNIST and SVHN are not studied here since other related work [11, 17] can already obtain close or even no accuracy loss.

4.1. CIFAR-10

To impartially compare our ternarization method with [18], we choose the same ResNet-20, 32, 44, 50 (type-A parameter free residual connection) [7] as the testing neural network architecture on CIFAR-10 dataset. CIFAR-10 contains 50 thousands training samples and 10 thousands test samples with 32×32 image size. The data augmentation method is identical as used in [7]. For fine-tuning, we set the initial learning rate as 0.1, which is scheduled to scale by 0.1 at epoch 80, 120 and 160 respectively. Note that, both the first and last layer are ternarized during the training and test stage.

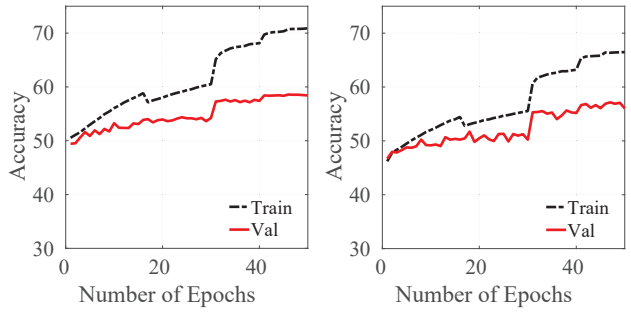
we report the CIFAR-10 inference accuracy as listed in Table 1. It can be seen that only ternarized ResNet-20 gets minor (0.64%) accuracy degradation, while all the other deeper networks like ResNet-32, 44, 56 actually outperform the full precision baseline with 0.12%, 0.24% and 0.18%, respectively. It shows that a more compact neural network, like ResNet20, is more likely to encounter accuracy loss, owing to the network capacity is hampered by this aggressive model compression. Similar trend is also reported in TTN [18]. Meanwhile, except ResNet56, other accuracy improvements of our method are higher. Note that, the improvement is computed as the accuracy gap between pre-trained full precision model and corresponding ternarized model. Our pre-trained full precision model accuracy is slightly different even with the same configurations.

Table 1. Inference accuracy of ResNet on CIFAR-10 dataset

		ResNet20	ResNet32	ResNet44	ResNet56
TTN [18]	FP	91.77	92.33	92.82	93.2
	Tern	91.13	92.37	92.98	93.56
	Gap	-0.64	+0.04	+0.16	+0.36
our work	FP	91.7	92.36	92.47	92.68
	Tern	91.65	92.48	92.71	92.86
	Gap	-0.05	+0.12	+0.24	+0.18

4.2. ImageNet

In order to provide a more comprehensive experimental results on large dataset, we examine our model ternarization techniques on image classification task with ImageNet [14] (ILSVRC2012) dataset. ImageNet contains 1.2 million training images and 50 thousands validation images, which are labeled with 1000 categories. For the data pre-processing, we choose the scheme adopted by ResNet [7]. Augmentations applied to the training images can be sequentially enumerated as: 224×224 randomly resized crop, random horizontal flip, pixel-wise normalization. All the reported classification accuracy on validation dataset is single-crop result. Beyond that, similar as other popular model quantization works, such as XNOR-Net [13], DoreFa-Net [17] and TTN [18], we first keep the first and last layer in full precision (i.e. 32-bit float) and ternarize the remaining layers for fair comparison. In addition of that, we also conduct experiments to fully ternarize the whole network. Thus in this case, the weight parameters of the whole network is in ternary format.



(a) first&last layer FP (b) first&last layer Tern.

Figure 3. Train and validation (top-1) accuracy of AlexNet on ImageNet

Table 2. Validation accuracy of AlexNet on ImageNet using various model quantization methods. FP, Bin., Tern. denote full-precision, binary and ternary respectively.

	Quantize scheme	First layer	Last layer	Accuracy (top1/top5)	Comp. rate
DoreFa-Net [17, 18]	Bin.	FP	FP	53.9/76.3	∼32×
BWN [13]	Bin.	FP	FP	56.8/79.4	∼32×
ADMM [10]	Bin.	FP*	FP*	57.0/79.7	∼32×
TWN [10, 11]	Tern.	FP	FP	57.5/79.8	∼16×
ADMM [10]	Tern.	FP*	FP*	58.2/80.6	∼16×
TTN [18]	Tern.	FP	FP	57.5/79.7	∼16×
Full precision	-	FP	FP	61.78/82.87	1×
this work	Tern.	FP	FP	58.59/80.44	∼16×
this work	Tern.	Tern	Tern	57.15/79.42	∼16×

4.2.1 AlexNet:

We first present the experimental results and its analysis on AlexNet, which is similar as the variant AlexNet model used by DoreFa-Net [17] and TTN [18]. The AlexNet in this work does not include Local-Response-Normalization (LRN) [8]. We apply a batch normalization layer after each convolution and fully-connected layer. The dropout layer is removed, since we find that it actually hampers the performance of fine-tuning of ternarized model from its full precision counterpart. A possible reason is that dropout method randomly deactivates some weighted connections. However our kernel scaling factor is calculated with respect to the weight distribution of entire layer.

We first train the full precision AlexNet with the aforementioned data augmentation method. SGD with momentum = 0.9 is taken as the optimizer. The initial learning rate is 0.1 which is scheduled with decay rate of 0.1 at epoch 45 and 65. Batch size is set to 256, and weight decay is 1e-4. For fine-tuning ternarized AlexNet from the pre-trained full-precision model, we alter the optimizer to Adam. The initial learning rate is set to 1e-4 and scaled by 0.2, 0.2 and 0.5 at epoch 30, 40 and 45 respectively. Batch size is 256 as the full-precision model. Since low-bit quantization could also be considered as a strong regularization techniques, weight decay is expected to be at small value or even zero. Here we set the weight decay as 5e-6.

We here report the accuracy on ImageNet validation set including our work and other recently published works with the state-of-the-art performance in Table 2. Note that, we mark out the quantization state of the first layer and last layer for various methods. For works without such discussion in the paper and no released code, we consider that they configure the first and last layer in full precision (marked with *). We are confident in such assumption since we double checked the works they compared in their published papers are with first- and last-layer in full precision. The results show that our work achieves the best accuracy compared to most recent works. In comparison to the previous best result reported by ADMM [10], our work improves the ADMM binary scheme and ternary scheme by 1.59% and 0.39% respectively.

Furthermore, different from previous works that all preserve the weights of first and last layer in full-precision, we also conduct experiments to fully ternarize the whole network. Such ternarization leads to a further 1.44% top-1 accuracy drop. For a general purpose processor, like CPU/GPU, the performance improvement by ternarizing the whole network is not quite huge since the parameter size reduction and computation cost reduction are relatively small. However, for a domain-specific neural network accelerator, like ASIC and FPGA, ternarizing the entire network means no specific convolution cores are needed to perform Multiplication and Accumulation (MAC) operations [2]. It will save large amount of power, area and other on-chip hardware resources, which further improve the system performance. We will provide more quantitative analysis about this issue in the later discussion section.

4.2.2 ResNet:

Various works [16, 18] have drawn a similar conclusion that, under the circumstance of using identical model compression technique, accuracy degradation of the compressed model w.r.t its full-precision baseline will be enlarged when the model topology becomes more compact. In other words, neural network with smaller model size is more likely to encounter accuracy degradation in weight compression. Meanwhile, since neural network with residual connections is the main stream of deep neural network structure, we also employ the compact ResNet-18 (type-b residual connection in [7]) as another benchmark to demonstrate that our ternarization method could achieves the best state-of-the-art performance. The full-precision ResNet18 model [2] released by pytorch framework is used as the baseline and pretrained model in this work. The data augmentation method is same as we introduced in Section 4.2.1 for AlexNet.

[2]https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

Table 3. Validation accuracy (top1/top5 %) of ResNet-18b [7] on ImageNet using various model quantization methods.

	Quan. scheme	First layer	Last layer	Accuracy (top1/top5)	Comp. rate
BWN [13]	Bin.	FP	FP	60.8/83.0	∼32×
ABC-Net [12]	Bin.	FP*	FP*	68.3/87.9	∼6.4×
ADMM [10]	Bin.	FP*	FP*	64.8/86.2	∼32×
TWN [10, 11]	Tern.	FP	FP	61.8/84.2	∼16×
TTN [18]	Tern.	FP	FP	66.6/87.2	∼16×
ADMM [10]	Tern.	FP*	FP*	67.0/87.5	∼16×
Full precision	-	FP	FP	69.75/89.07	1×
this work	Tern.	FP	FP	67.95/88.0	∼16×
this work	Tern.	Tern	Tern	66.01/86.78	∼16×

As the simulation results listed in Table 3, ResNet-18b [7] with our ternarization scheme shows 0.95% and 0.5% higher top-1 and top-5 accuracy, respectively, compared to previous best result of ADMM [10] with equal compression rate (16×). ABC-Net [12] has $\sim$ 0.3% higher top-1 accuracy over our work, but with a cost of 2.5 times larger model size. Moreover, our work save ∼60% computation owing to the sparse kernel after ternarization. Similar as in AlexNet, if we further ternarize the first and last layer, both the top-1 and top-5 accuracy of ResNet-18 degrades.

Table 4. Inference accuracy (Top1/Top5 %) of ternarized ResNets on ImageNet dataset

	ResNet18	ResNet34	ResNet50	ResNet101
FP	69.75/89.07	73.31/91.42	76.13/92.86	77.37/93.55
Tern	66.01/86.78	70.95/89.89	74.00/91.77	75.63/92.49
Gap	-3.74/-2.29	-2.36/-1.53	-2.13/-1.09	-1.74/-1.06

Moreover, we conduct a series of deeper residual networks (ResNet-34, 50, 101) in addition to the ResNet-18. As listed in Table 4, the accuracy gap between a ternarized model and its full-precision counterpart is reduced when network goes deeper. Such observation is in consistent with the experiments on the CIFAR-10 dataset.

4.2.3 Improve accuracy on ResNet with REL

We introduce the Residual Expansion Layer (REL) technique to compensate the accuracy loss, which can still benefit from the weight ternarization in model size and computation reduction, and the multiplication elimination for convolution and fully-connected layers. In order to show that such residual expansion can effectively reduce the accuracy gap between ternarized model and full-precision baseline, we report the validation accuracy of expanded residual ResNet18 on ImageNet in Table 5. When expansion factor T_{ex} is 2 and two thresholds $\beta = \{0.05, 0.1\}$ are used, we succeeded to further shrink the accuracy gap to -1.7%/-1.03%. When the T_{ex} is increased to 4 ($\beta = \{0.05, 0.1, 0.15, 0.2\}$), the accuracy gap is almost negligible.

Table 5. Validation accuracy (top1/top5 %) of ResNet-18b on ImageNet with/without residual expansion layer (REL).

	First layer	Last layer	Accuracy (top1/top5)	Accuracy gap	Comp. rate
Full precision	FP	FP	69.75/89.07	-/-	1×
T_{ex}=1	FP	FP	67.95/88.0	-1.8/-1.0	∼ 16×
T_{ex}=1	Tern	Tern	66.01/86.78	-3.74/-2.29	∼ 16×
T_{ex}=2	FP	FP	**69.33/89.68**	**-0.42/+0.61**	∼ 8×
T_{ex}=2	Tern	Tern	68.05/88.04	-1.70/-1.03	∼ 8×
T_{ex}=4	Tern	Tern	**69.44/88.91**	**-0.31/-0.16**	∼ 4×

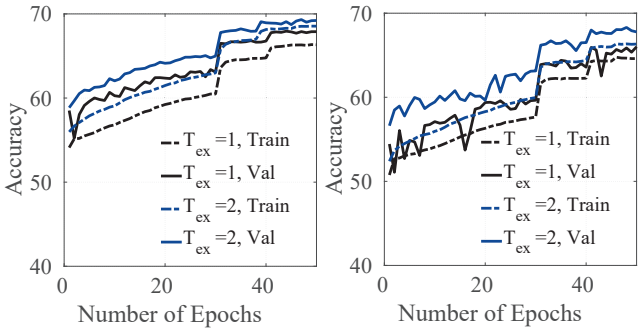
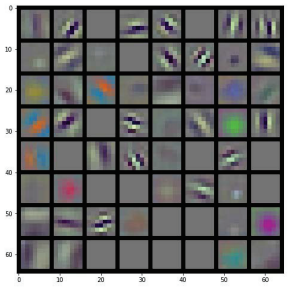


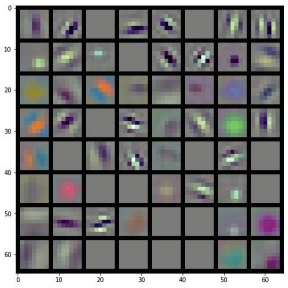
Figure 4. The accuracy evolution curve of ternarized ResNet-18b on ImageNet with residual expansion. (a) First and last layer in full-precision; (b) First and last layer are ternarized

The training and test accuracy evolution accuracy of ResNet18 with all the configurations listed in Table 5 are plotted in Fig. 4. Based on our observation, whether implementing ternarization of the first and last layer does have a significant effect on the training process and accuracy. Ternarizing the first and last layer of ResNet18 (Fig. 4b) causes large validation accuracy fluctuation during training compared to the experiments that keep the first and last layer in full precision (Fig. 4a). As shown in Fig. 4b, the validation accuracy curve with residual expansion alleviate the fluctuation and smooth the curve.

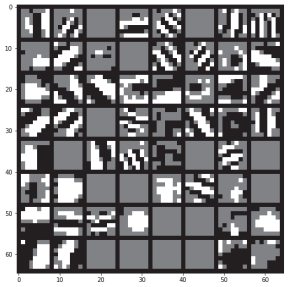
Beyond that, we further explore the effect of ternarization process of neural network with the assistance of kernel visualization. Since we also ternarize the first convolution layer of ResNet18, the input channel-0 is used as an example to study the weight transformation during weight ternarization as shown in Fig. 5. Kernels plotted in Fig. 5a are from the full-precision pretrained model, which are taken as the initial weights. After fine-tuning the model for weight ternarization, the full-precision weights (i.e., the full precision weight is kept for back-propagation) shown in Fig. 5b still keep the similar patterns as in the pretrained model. During inference in validation stage, kernels in Fig. 5b are ternarized with two different threshold factors, $\beta = 0.05$ and $\beta = 0.1$ for the original layer *Conv* and its REL *Conv*$_r$



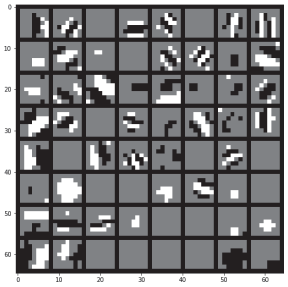
(a) Full-precision kernel before fine-tuning



(b) Full-precision kernel after fine-tuning



(c) Ternarized kernel with threshold factor $\beta = 0.05$.



(d) Ternarized kernel with threshold factor $\beta = 0.1$.

Figure 5. Kernel visualization for the first layer (channel 0) of ResNet18 before and after ternarization.

respectively. As seen in Fig. 5c and Fig. 5d, both ternarized kernels, to some extent, preserve the features from full precision model. Ternarized kernels with larger threshold factor only keep the connection with higher weight intensity. Thus, the residual expansion layer with higher threshold factor will lead to higher layer sparsity, and only a very small portion weights (i.e., non-zero value) will use the computing resources to perform network inference.

5. Ablation study and discussion

5.1. Ablation study

Table 6. Validation accuracy of ResNet-18b on ImageNet.

	Accuracy	Comp. rate
Baseline: Full precision	69.75/89.07	1×
TW	57.48/81.15	∼16×
TW+ICS ($\beta = 0.05$)	65.08/86.06	∼16×
TW+ICS ($\beta = 0.1$)	64.78/86.06	∼16×
TW+ICS ($\beta = 0.2$)	57.09/81.01	∼16×
TW+FT	64.94/86.13	∼16×
TW+ICS+FT ($\beta = 0.05$)	66.01/86.78	∼16×
TW+ICS+FT+REL ($\beta = \{0.05, 0.1\}$)	68.05/88.04	∼8×

In order to show the effectiveness of the proposed methods, we present a fair ablation study (Table 6) that isolates multiple factors which are listed as follow: 1) TW: directly training the Ternarized Weight (TW) from scratch without Iteratively Calculated Scaling factors (ICS); 2) TW+ICS: Training the ternarized network from scratch with the varying scaling factor (β=0.05, 0.1 and 0.2); 3) TW+FT: fine tuning (FT) the ternarized weight. It shows that using ICS and training from scratch is close to the performance of fine-tuning the model without ICS. 4) TW+ICS+FT: Fine-tuning the the model with ICS further improves the accuracy. 5) TW+ICS+FT+REL: incorporating our proposed REL techniques leads to almost no accuracy loss.

5.2. Deep CNN density examination

Table 7. Density variation of first and last layer before/after fine-tuning

	First layer		Last layer		Average density
	Before	After	Before	After	
AlexNet	36.3%	53.6%	62.8%	67.5%	41.1%
ResNet18	41.7%	52.2%	54.8%	68.3%	40.1%

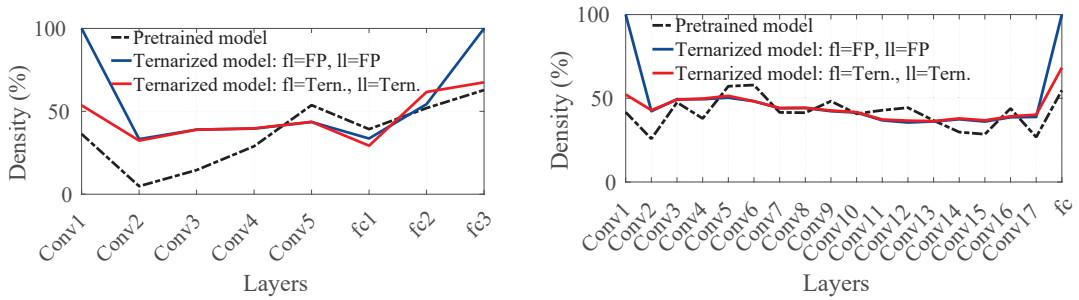
Let us assume the trained sparsity (sparsity=1-density) of the designated layer is correlated to its redundancy. Fig. 6 depicts the weight density distribution of all the convolution and fully-connected layers in AlexNet and ResNet for the cases of before and after the model ternarization. As shown in both Fig. 6a and Fig. 6b, the weight density of first and last layer is raised to a higher level after ternarization in comparison to their initial density. Here the density on pre-trained model refers to the neural network that initialized from the full-precision pretrained model, then we directly apply our ternarization method as introduced in Section 3 without fine-tuning the weights. We calculate the average density of all the convolution and fully-connected layers, which are reported in Table 7. It shows that after ternarization, the density of first and last layer is much higher than the average density and other layers. A potential method to further compress network model is to use the density as an indicator to choose different expansion factors for each each layer.

5.3. REL versus multi-bit weight

Table 8. Inference accuracy of ResNet-18 on ImageNet with multi-bit [17] weight or REL.

	Multi-bit [17]	REL (this work)	
Config.	3-bit	t_{ex}=1	t_{ex}=2
Accuracy	65.70/86.82	66.01/86.78	68.05/88.04
Comp. rate	10.7×	16×	8×

In comparison to directly using multiple bits (multi-bit) weight, using REL is advantageous in terms of both accuracy and hardware implementations. First, REL shows better performance over the multi-bit method adopted from Dorefa-Net [17] on ImageNet. Second, from the hardware perspective, REL preserves all network weights in the ternary format (i.e. -1, 0, +1), enabling the implementation of convolutions through sparse addition operations without multiplication. Increasing the number of expansion layers



(a) AlexNet (b) ResNet-18

Figure 6. Density distribution of convolution and fully-connected layers in (a) AlexNet and (b) ResNet-18 structure. The weight density of pretrained model is when the network just load the parameters from pretrained model without fine-tuning. Ternarized model: fl=FP, ll=FP refers the density distribution of the ternarized model with first and last layer in full precision. Ternarized model: fl=Tern., ll=Tern. denotes that the density distribution of the ternarized model with first and last layer in ternary representation.

(t_{ex}) merely requires more convolution kernels when one deploys the network in FPGA/ASIC. It is noteworthy that the REL with larger threshold factor owns higher sparsity, which requires less computational resources. On the contrary, when more bits are added in multi-bit, the convolution computation goes back to multiplication and addition operations, consuming more energy and chip area. Finally, both our work (Section 5.2) and the deep compression [6] show that a deep neural network has different redundancies from layer to layer. If one employs the multi-bit scheme, we have to re-design the computing hardware with different bit-widths for the layers. Beyond that, since zero-value weights in REL with smaller threshold will be stay in zero in REL with larger threshold (as shown in Fig. 5c and Fig. 5d), the model compression rate can be further improved with proper weight sparse encoding method.

5.4. Hardware resource utilization efficiency

Table 9. FPGA resource utilization (Kintex-7 XC7K480T) for AlexNet last layer

Weight type	MACs		LUT (slices)		DSP (slices)	
	Multiplier	Adder	required	available	required	available
FP	100	100	49600	74650	200	1920
Tern.	0	90	23490	74650	0	1920

As we previously discussed in Section 4.2.1, fully ternarizing the first and last layer is greatly beneficial for developing the corresponding hardware accelerator on FPGA or ASIC, owing to the elimination of floating point convolution computing core. Here in this work, we analyze the deployment of the last fully connected layer of AlexNet into a Xilinx high-end FPGA (Kintex-7 XC7K480T) as an example. As reported by [5], floating-point adder cost 261 LUT slices on Kintex-7 , while the floating-point multiplier takes 235 LUT slices with additional 2 DSP48Es. The weights of last fully-connected layer in AlexNet is in the dimension of 4096×1000, which requires 1000 *MACs* [3] if performing the computation in the extreme parallel manner. However, due to the FPGA resource limitation, we consider the design scheme that allows 100 MACs for floating-point weight convolution, while the rest resources can only accommodate 90 MACs for ternary weight convolution. For the neural network with both ternary and full precision weights, balancing the hardware resources to build MAC cores for ternary and full precision weight becomes extremely difficult. Increasing the number of ternary weight MAC cores does improve the hardware resource utilization. However, it has to reduce the allocated floating point MACs, which at the same time reduces the computation parallelism and thus increases whole system latency. On the contrary, if we keep the current hardware resource allocation plan, the hardware utilization efficiency will be extremely low especially when the networks are going deeper. Then, it can be clearly seen that fully ternarizing the whole network is extremely important for algorithm deployment in resource limited embedded system although the theoretical model compression rate does not increase too much.

6. Summary

In this work, we have explicitly optimize current DNN ternarization scheme with techniques like statistical weight scaling and REL. On image classification task, a series of comprehensive experiments are conducted to show that our method can achieve the state-of-the-art accuracy on both small dataset like CIFAR-10, and large dataset like ImageNet. Beyond that, we examined the accuracy with/without the first&last layer ternarization. Owing to that the statistical weight scaling are used through the entire network, our ternarized network does not encounter serious accuracy degradation even with ternarized weight for first&last layer.

Acknowledgement This work is supported in part by the National Science Foundation under Grant No. 1740126 and Semiconductor Research Corporation nCORE.

References

[1] Y. Bengio et al. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*, pages 262–263. IEEE, 2016.

[3] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.

[4] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[5] A. Ehliar. Area efficient floating-point adder and multiplier with ieee-754 compatible semantics. In *Field-Programmable Technology (FPT), 2014 International Conference on*, pages 131–138. IEEE, 2014.

[6] S. Han et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[10] C. Leng, H. Li, S. Zhu, and R. Jin. Extremely low bit neural network: Squeeze the last bit out with admm. *arXiv preprint arXiv:1707.09870*, 2017.

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

[12] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 344–352, 2017.

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

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[15] W. Sung, S. Shin, and K. Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015.

[16] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, pages 2625–2631, 2017.

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

[18] C. Zhu et al. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.