N-ARY QUANTIZATION FOR CNN MODEL COMPRESSION AND INFERENCE ACCELERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The tremendous memory and computational complexity of Convolutional Neural Networks (CNNs) prevents the inference deployment on resource-constrained systems. As a result, recent research focused on CNN optimization techniques, in particular quantization, which allows weights and activations of layers to be represented with just a few bits while achieving impressive prediction performance. However, aggressive quantization techniques still fail to achieve full-precision prediction performance on state-of-the-art CNN architectures on large-scale classification tasks. In this work we propose a method for weight and activation quantization that is scalable in terms of quantization levels (n-ary representations) and easy to compute while maintaining the performance close to full-precision CNNs. Our weight quantization scheme is based on trainable scaling factors and a nested-means clustering strategy which is robust to weight updates and therefore exhibits good convergence properties. The flexibility of nested-means clustering enables exploration of various n-ary weight representations with the potential of high parameter compression. For activations, we propose a linear quantization strategy that takes the statistical properties of batch normalization into account. We demonstrate the effectiveness of our approach using state-of-the-art models on ImageNet.

1 INTRODUCTION

The increasing computational complexity and memory requirements of Convolutional Neural Networks (CNNs) have motivated recent research efforts in efficient representation and processing of CNNs. Several optimization and inference approaches have been proposed with the objective of model compression and inference acceleration. The primary aim of model compression is to enable on-device storage (e.g., mobile phones and other resource-constrained devices) and to leverage on-chip memory in order to reduce energy consumption (Horowitz, 2014), latency and bandwidth of parameter accesses. Inference acceleration can be achieved by lowering the precision of computations in terms of resolution, removing connections within networks (pruning), and specialized software/hardware architectures.

Quantized Neural Networks are optimization techniques where weights and/or activations of a neural network are transformed from 32-bit floating point into a lower resolution; for aggressive quantization techniques down to binary (Courbariaux et al., 2015; Rastegari et al., 2016) or ternary (Li & Liu, 2016; Zhu et al., 2016) representations. Although prior quantization techniques achieve full-precision accuracy on highly over-parameterized architectures (e.g., AlexNet, VGG) or toy tasks (e.g., MNIST, SVHN, CIFAR-10), there is still an unacceptable gap between extremely low bit-width representations and state-of-the-art architectures for real-world tasks. Furthermore, aggressive quantization approaches are usually designed for specific representations (e.g., binary or ternary), and are not scalable in the sense that they do not allow for more quantization levels (i.e., weight values) if required. Thus, accuracy degradation cannot be compensated without changes in the baseline architecture which includes deepening or widening the neural network, and/or using full-precision weights in the input and output layers, respectively.

In this work, we address the issue of accuracy degradation by introducing a scalable non-uniform quantization method for weights that is based on trainable scaling factors in combination with a

nested-means clustering approach. In particular, nested-means splits the weight distribution iteratively into several quantization intervals until a pre-defined discretization level is reached. Subsequently, all weights within a certain quantization interval are assigned the same weight (i.e., the scaling factor). Nested-means clustering tends to assign small weights to larger quantization intervals while less frequent larger weights are assigned to smaller quantization intervals. This improves classification performance which is in line with recent observations that larger weights carry more information than smaller weights (Han et al., 2015a). We evaluate our approach on state-of-the-art CNN architectures in terms of computational requirements and prediction accuracy using the ImageNet classification task.

The paper is structured as follows. In Sec. 2 related work is discussed. Weight and activation quantization is presented in Sec. 3 and Sec. 4, respectively. Experimental results for ImageNet are shown in Sec. 5. Sec. 6 concludes the paper.

2 Related Work

CNN quantization is an active research area with various approaches and objectives. In this section, we briefly review the most promising strategies from different categories. We distinguish between two orthogonal qualitative dimensions: (i) Approaches that only quantize the weights and approaches that quantize both the weights and the activations, and (ii) scalable approaches that allow for different bit-widths and non-scalable approaches that are designed for specific weight representations.

Non-scalable weight-only quantization: Courbariaux et al. (2015) introduced Binary Connect (BC) in which they constrain weights to either -1 or 1. Rastegari et al. (2016) proposed Binary Weight Networks (BWNs), which improves BC by introducing channel-wise scaling factors. Li & Liu (2016) introduced Ternary Weight Networks (TWNs), in order to account for the accuracy degradation of BWNs. Zhu et al. (2016) proposed Trained Ternary Quantization (TTQ), extending TWNs by non-uniform and trainable scaling factors.

Scalable weight-only quantization: Han et al. (2015a) proposed Deep Compression, a method that leverages pruning, weight sharing and Huffman Coding for model compression. Zhou et al. (2017) proposed Incremental Network Quantization (INQ) to quantize pre-trained full-precision DNNs to zeros and powers of two. This is accomplished by iteratively partitioning the weights into two sets, one of which is quantized while the other is retrained to compensate for accuracy degradation.

Non-scalable quantization for weights and activations: Binarized Neural Networks (Courbariaux & Bengio, 2016) and XNOR-Net (Rastegari et al., 2016) quantize both weights and activations to either -1 or 1. Cai et al. (2017) proposed low-precision activations using Half-Wave Gaussian Quantization (HWGQ), and relies on BWN for weight binarization. Based on symmetric binary and ternary weights as proposed by Zhu et al. (2016), Faraone et al. (2018) introduced Symmetric Quantization (SYQ) by pixel-wise scaling factors and fixed-point activation quantization.

Scalable quantization for weights and activations: DoReFa-Net (Zhou et al., 2016) allows weights and activation to be variable configurable in the length through fixed-point quantization. Baskin et al. (2018) proposed uniform noise injection for non-uniform quantization (UNIQ) of both weights and activations. Zhang et al. (2018) proposed an adaptively learnable quantizer (LQ-Nets) and achieves state-of-the-art accuracy as of today.

In this work, we focus on a scalable quantization of weights and activations, with the main objective of maintaining accuracy while exploiting model compression and reduced computational complexity for inference acceleration. Our approach differs from prior work as we introduce a novel quantization strategy that is easy to compute, maintains state-of-the-art prediction accuracy, and has significantly less impact on training time. In particular, our approach enables various non-uniform n-ary weight representations with a high amount of sparsity.

3 Weight Quantization

For weight quantization, we employ the common strategy of maintaining full-precision weights for training that are quantized during forward propagation. The gradient of the full-precision weights

is approximated by backpropagating through quantization functions using the straight-through gradient estimator (STE) (Bengio et al., 2013). The gradient is subsequently used to update the full-precision weights. For inference, the full-precision weights are discarded and only the quantized weights are used for model compression and inference acceleration. In this section we describe the quantization strategy that is used at forward propagation.

3.1 THRESHOLDING AND SCALING FACTORS

The classification accuracy of aggressive quantization techniques heavily relies on the usage of scaling factors as model capacity is improved significantly. For binary weights, Rastegari et al. (2016) propose one uniform scaling factor $\alpha_{l,k}$ per layer l and output feature map k, which is calculated as the mean of absolute floating-point weights, i.e., $\alpha_{l,k} = (\sum_{w \in \mathbf{W}_{l,k}} |w|)/|\mathbf{W}_{l,k}|$ where $\mathbf{W}_{l,k}$ denotes the set of weights connected to output feature map k. For ternary weights, Zhu et al. (2016) propose two trainable non-uniform scaling factors α_l^p and α_l^n per layer that are determined by gradient descent. This adjusts the scaling factors so as to minimize the given loss function while at the same time increasing the model capacity due to non-uniform scaling. Therefore, we adopt trainable scaling factors in our method. Let $0 < \delta_{l,1}^p < \ldots < \delta_{l,K_p}^p$ and $0 < \delta_{l,1}^n < \ldots < \delta_{l,K_n}^n < \delta_{l,K_{n+1}}^n$ be two sets of interval thresholds for positive and negative weights, respectively, that partition the real numbers into intervals

$$\mathbf{\Delta}_{l,K_n}^n = (-\infty, -\delta_{l,K_n}), \ \mathbf{\Delta}_{l,i}^n = \left[-\delta_{l,i+1}^n, -\delta_{l,i}^n\right), \ \mathbf{\Delta}_{l,i}^p = \left[\delta_{l,i}^p, \delta_{l,i+1}^p\right), \ \mathbf{\Delta}_{l,K_p}^p = \left[\delta_{l,K_p}^p, \infty\right). \tag{1}$$

If the zero weight should be explicitly modeled, we define an interval $\mathbf{\Delta}_{l,0} = [-\delta_{l,1}^n, \delta_{l,1}^p)$. If no explicit zero weight is modeled, we define two intervals $\mathbf{\Delta}_{l,0}^n = [-\delta_{l,1}^n, 0)$ and $\mathbf{\Delta}_{l,0}^p = [0, \delta_{l,1}^p)$. To each positive interval $\mathbf{\Delta}_{l,i}^p$, we assign a trainable scaling factor $\alpha_{l,i}^p$ that is used to quantize the weights as $w^q = \alpha_{l,i}^p \Leftrightarrow w \in \mathbf{\Delta}_{l,i}^p$. For negative intervals, we analogously have $w^q = -\alpha_{l,i}^n \Leftrightarrow w \in \mathbf{\Delta}_{l,i}^n$. During training, we update the scaling factors $\alpha_{l,i}^p$ and $\alpha_{l,i}^n$ using gradients computed as

$$\frac{\partial E}{\partial \alpha_{l,i}^p} = \sum_{j:w_j \in \mathbf{\Delta}_{l,i}^p} \frac{\partial E}{\partial w_{l,j}^q}, \quad \frac{\partial E}{\partial \alpha_{l,i}^n} = \sum_{j:w_j \in \mathbf{\Delta}_{l,i}^n} \frac{\partial E}{\partial w_{l,j}^q}, \tag{2}$$

where E denotes the loss function and w_l^q denotes the quantized weights. In case the zero weight is modeled, we have a fixed scaling factor $\alpha_{l,0} = 0$ that is not updated during gradient descent. Finding good interval thresholds δ is essential for prediction performance and will be discussed in Sec. 3.3.

3.2 NON-UNIFORM WEIGHT REPRESENTATIONS

Allowing the weights to be non-uniform enables various explorations of weight representations. Weight distributions tend to be Gaussian with an arithmetic mean close to zero (see Fig. 1). Therefore, a good approximation of the weight distribution are representations that are symmetric around zero. Candidates for such representations are summarized in Table 1.

Table 1: Different non-uniform n-ary weight representations

Notation	Representation	Bit width (dense)	Bit width (sparse)
Binary	$\{\alpha_0^p, \alpha_0^n\}$	1	–
Ternary	$\{\alpha_0^p, 0, \alpha_0^n\}$	2	1
Quaternary	$\{\alpha_1^p, \alpha_0^p, \alpha_0^n, \alpha_1^n\}$	2	–
Quaternary+	$\{\alpha_1^p, \alpha_0^p, 0, \alpha_0^n\}$	2	2
Quaternary-	$\{\alpha_0^p, 0, \alpha_0^n, \alpha_1^n\}$	2	2
Quinary	$\{\alpha_1^p, \alpha_0^p, 0, \alpha_0^n, \alpha_1^n\}$	3	2

Binary and ternary representations gained a lot of interest lately due to their high compression ratios and, in view of their low expressiveness, relatively good prediction performance. On large-scale

classification tasks, however, only ternary weights demonstrate a prediction accuracy similar to full-precision weights. Furthermore, for ternary representations weight pruning is possible with little impact on prediction accuracy which additionally reduces the computational complexity of inference. In this work, we also explore other n-ary weight representations that facilitate compression levels similar to ternary weights while significantly improving model capacity. For instance, quaternary weights can also be encoded with only two bits, but introduce either an additional positive (quaternary+) or negative (quaternary-) scaling factor, respectively. Quinary weights extend ternary weights by one positive and one negative value, but are still encoded with only two bits in a sparse format. In a sparse format, only the indices of non-zero weight entries are stored such that two bits are sufficient to represent the non-zero values.

3.3 Nested-Means Clustering

For an optimal approximation, weight clustering is required to partition a set of weights that are later represented by a single discrete value per cluster (cf. Sec. 3.1). The clustering can be implemented either statically (once before training) or dynamically (repeatedly during training) by calculating thresholds δ which represent the boundaries of the respective cluster.

The static approach has the advantage of allowing iterative clustering algorithms to be applied (e.g., k-means clustering or Lloyd's algorithm (Lloyd, 1982)) that are able to find an optimal solution for the cluster assignment. However, as quantization lowers the resolution and therefore changes in the weight distribution, quantization requires re-training to compensate for this loss of information. As a consequence, the optimal solution found by an iterative algorithm will become non-optimal during the following re-training process. Lowering the learning rate for re-training can diminish heavy changes in the weight distribution, at the cost of longer time to converge and the risk to get stuck at plateau regions, which is especially critical for trainable scaling factors (Eq. 2). Applying an iterative clustering approach repeatedly during training is practically infeasible, since it causes a dramatic increase in training time.

A practical useful clustering solution is to calculate cluster thresholds during training based on the maximum absolute value of the full-precision weights per layer (Zhu et al., 2016): $\delta_l = t \cdot \max(|w_l|)$, where t is a hyperparameter. This approach is beneficial because it defines cluster thresholds which are influenced by large weights that were shown to play a more important role than smaller weights (Han et al., 2015b). Furthermore, training time is virtually unaffected by this rather simple calculation. However, having an additional hyperparameter t_i for each scaling factor α_i renders the mandatory hyperparameter tuning infeasible. Furthermore, the sensitivity to the maximum value results in aggressive threshold changes caused by weight updates, possibly even preventing the network from converging.

In order to overcome these issues, we propose a symmetric nested-means clustering algorithm for assigning full-precision weights to a set of quantization clusters. Since weight distributions are typically Gaussian with a mean close to zero, we divide the weights into a negative and a positive cluster ($\mathbf{I}^p_{l,1}$ and $\mathbf{I}^n_{l,1}$) per layer l. These clusters are then divided at their arithmetic means $\delta^p_{l,i}$ and $-\delta^n_{l,i}$ into two subclusters – for each cluster we obtain an inner cluster and an outer cluster containing the tail of the distribution. The subclusters containing the tail of the distribution ($\mathbf{I}^p_{l,i}$ and $\mathbf{I}^n_{l,i}$) are repeatedly divided at their arithmetic means until the targeted number of quantization intervals is reached. More formally, nested-means clustering iteratively computes

$$\mathbf{I}^p_{l,i} = \{j | w_{l,j} > \delta^p_{l,i-1}\} \quad \text{and} \quad \mathbf{I}^n_{l,i} = \{j | w_{l,j} < -\delta^n_{l,i-1}\} \tag{3}$$

$$\delta^p_{l,i} = \frac{1}{|\mathbf{I}^p_{l,i}|} \sum_{j \in \mathbf{I}^p_{l,i}} |w_{l,j}| \quad \text{and} \quad \delta^n_{l,i} = \frac{1}{|\mathbf{I}^n_{l,i}|} \sum_{j \in \mathbf{I}^n_{l,i}} |w_{l,j}|, \tag{4}$$

starting from $i = 1$ with $\delta^p_{l,0} = \delta^n_{l,0} = 0$. The whole clustering process is shown in Fig. 1 on the example of seven quantization clusters.

The arithmetic mean is consistent with the centralization demands of the cluster thresholds due to its central tendencies. At the same time the mean is less sensitive to weight updates than the maximum of weight values, allowing for better convergence. Last, nested-means clustering requires only one arithmetic mean per cluster, which is computationally efficient and avoids the need for additional hyperparameters.

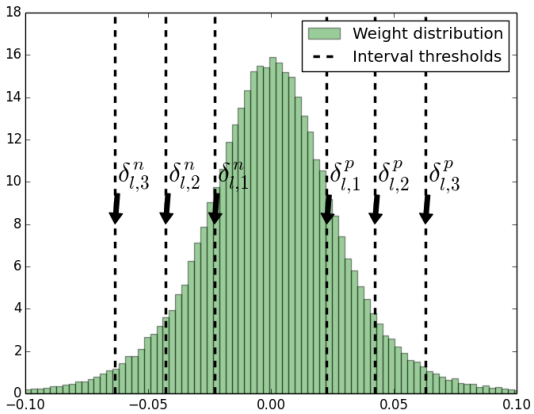

Figure 1: Nested-means intervals of a trained ResNet layer.

4 Activation Quantization

The rectified linear unit (ReLU) activation function $f(x) = \max(0, x)$ is a commonly used non-linear activation function due its computational efficiency and its ability to alleviate the vanishing gradient problem. However, the ReLU function produces unbounded outputs which potentially require a high dynamic range and are therefore difficult to quantize. A common solution for this is to clip the outputs in the interval $(0, \gamma]$:

$$Q_c(x) = \begin{cases} 0 : x \leq 0 \\ x : 0 < x \leq \gamma \\ \gamma : \gamma < x \end{cases} . \tag{5}$$

When selecting the clipping parameter γ, a trade-off needs to be made: On the one side, small values of γ produce gradient mismatches due to different forward and backward approximations in the clipped interval (γ, ∞). On the other side, large values of γ result in a large interval $(0, \gamma]$ that needs to be quantized. This is problematic if only a few bits are used as quantization errors might become large.

In order to define an appropriate clipping interval, we use the observation that pre-activations tend to have a Gaussian distribution (Ioffe & Szegedy, 2015). In a Gaussian distribution, most values lie within a rather small range and there are only a few outliers that yield a high absolute range. For instance, 99.7% of the values lie within three standard deviations σ of the mean μ (Smirnov & Dunin-Barkovskiĭ, 1963). We find this empirical rule to be a good approximation to filter out outliers and define the clipping interval as $\gamma = \mu + 3\sigma$. The truncated activations are then uniformly quantized to k-bits:

$$Q_a(x) = \underbrace{\frac{\gamma}{2^k - 1}}_{\beta} \cdot \underbrace{\text{round}\left(\frac{2^k - 1}{\gamma} Q_c(x)\right)}_{\text{k-bit integer}}, \tag{6}$$

where $\beta = \gamma/(2^k - 1)$ is a constant scaling factor. We approximate the gradient of the quantization function with the STE as

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial Q_a}. \tag{7}$$

This approach approximates the ReLU function well but suffers from the drawback that μ and σ need to be repeatedly calculated during training. In recent years, batch normalization became a standard tool to accelerate convergence of state-of-the-art CNNs (Ioffe & Szegedy, 2015). Batch normalization transforms individual pre-activations to approximately have zero mean and unit variance across all data samples. Cai et al. (2017) experimentally showed that the pre-activation distribution after batch normalization are all close to a Gaussian with zero mean and unit variance. Therefore, we

propose to select a fixed clipping parameter $\gamma = 3$ as it results in a small quantization interval $(0, \gamma]$ while also keeping the number of clipped activations $x > \gamma$ small.

5 EXPERIMENTS

We applied the proposed quantization approach on ResNet (He et al., 2015) and a variant of the Inception network (Ioffe & Szegedy, 2015), trained on the ImageNet classification task (Russakovsky et al., 2015). We use TensorFlow (Abadi et al., 2016) and the Tensorpack library (Wu et al., 2016). For the ResNet network, the learning rate starts from 0.1 and is divided by 10 each 30×100 iterations, a weight decay of 0.0001, and a momentum of 0.9 is used. For the Inception network, we schedule learning rates following the configuration of Wu et al. (2016), a weight decay of 0.0001, and a momentum of 0.9. We use eight GPUs for training and a batch size of 64 per GPU. The quantized networks leverage initialization with pre-trained, full-precision parameters. We quantize all convolutional layers and fully-connected layers except the input and output layers, respectively, to avoid accuracy degradation (Zhou et al., 2016; Zhu et al., 2016; Faraone et al., 2018; Zhang et al., 2018).

5.1 WEIGHT QUANTIZATION

We evaluate our weight quantization on ternary, quaternary- and quinary representations (Table 1) as these representation have good compression/acceleration potential while achieving the best accuracy. Table 2 reports the validation accuracy of ResNet-18 and Inception-BN on the ImageNet task.

Table 2: Validation accuracy (Top1,Top5), increase in training time and sparsity for different weight representation of ResNet18 and Inception-BN on ImageNet.

	Weights	Activations	Training	Top1	Top5	Sparsity
ResNet-18	bits	bits		%	%	%
Baseline	32	32	1.0x	70.4	89.5	–
Quinary	3	32	2.0x	69.7	89.0	53
Quaternary-	2	32	1.6x	69.2	88.7	50
Ternary	2	32	1.2x	68.2	88.0	61
Inception-BN	bits	bits		%	%	%
Baseline	32	32	1.0x	73.1	91.4	–
Quinary	3	32	1.1x	71.7	90.3	54
Quaternary-	2	32	1.1x	70.8	89.8	52
Ternary	2	32	1.0x	69.6	89.1	58

The training time increases with increasing quantization levels because of the additional computations of interval thresholds (Eq. 3). While the impact on training time is negligible for the Inception-BN model, the ResNet model shows an increase of up to 2.0x. This is caused by a better GPU utilization of ResNet which makes the overall training time more sensitive to additional operations.

5.2 ACTIVATION QUANTIZATION

In this section, we evaluate the activation quantization using ResNet-18 on ImageNet. We use the clipped ReLU (Eq. 5) and set $\gamma = 3$ (as discussed in Sec. 4) for quantized activations, and we use the ReLU without clipping and quantization for 32-bit activations. Table 3 reports the validation accuracy and the increase in training time for several activation bit-widths. The training time is only influenced by the weight quantization, whereas the influence of activation quantization is neglibible. The efficiency of the activation quantization is also shown in Fig. 2. As can be seen, the learning curves of ternary weights and 32-bit, 8-bit and 4-bit activations are roughly identical which highlights the robustness of the proposed quantization approach. Only 2-bit activations result in a slight accuracy degradation, but also this case shows a stable learning behavior.

Table 3: Validation accuracy (Top1,Top5) and increase in training time for different activation bit-width of ResNet-18 on ImageNet.

Weights	Activations	Training	Top1	Top5
Ternary	32	1.2x	68.2	88.0
Ternary	8	1.2x	68.1	88.0
Ternary	4	1.2x	68.1	88.1
Ternary	2	1.2x	66.2	86.7

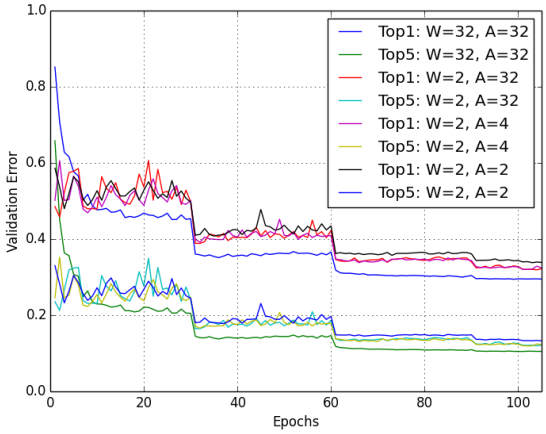

Figure 2: Validation error (Top1, Top5) of ResNet-18 on ImageNet for different bit-width combinations.

5.3 Comparison to Related Work

We compare the validation accuracy of different bit-width combinations of our approach to the most related work in Table 4. We also report the amount of sparsity and increase in training time (if it was reported by the authors). The results are obtained from the original publications with the exception of TTQ on Inception-BN, which we reproduced ourselves. Our method outperforms previous work for each bit-width combination of weights and activations. Furthermore, the simplicity of computing the nested-means clustering and the linear transformation of the activations is less computationally intensive and, therefore, results in less training time than previously reported results.

6 Conclusion

We have presented a novel approach for compressing CNNs through quantization and connection pruning, which reduces the resolution of weights and activations and is scalable in terms of the number of quantization levels. As a result, the computational complexity and memory requirements of DNNs are substantially reduced, and an execution on resource-constrained devices is more feasible. We introduced a nested-means clustering algorithm for weight quantization that finds suitable interval thresholds that are subsequently used to assign each weight to a trainable scaling factor. Our approach exhibits both a low computational complexity and robustness to weight updates, which makes it an attractive alternative to other clustering methods. Furthermore, the proposed quantization method is flexible as it allows for various numbers of quantization levels, enabling high compression rates while achieving prediction accuracies close to single-precision floating-point weights. For instance, we utilize this flexibility to add an extra quantization level to ternary weights (quaternary weights), resulting in an improvement in prediction accuracy while keeping the bit width at two. For activation quantization, we developed an approximation based on statistical attributes that have been observed when batch normalization is employed. Experiments using state-of-the-art DNN

Table 4: Comparison to related work (bold is ours) on the validation accuracy (Top1, Top5), increase in training time and sparsity of ResNet18 and Inception-BN on ImageNet.

	Weights	Activations	Training	Top1	Top5	Sparsity
ResNet-18	bits	bits		%	%	%
HWGQ (Cai et al., 2017)	1	2	–	59.6	82.2	–
SYQ (Faraone et al., 2018)	1	8	–	62.9	84.6	–
BWN (Rastegari et al., 2016)	1	32	–	60.8	83.0	–
LQ-Net (Zhang et al., 2018)	2	2	2.3x	64.9	85.9	–
Ours (ternary)	2	2	1.2x	**66.2**	**86.7**	61
Ours (ternary)	2	4	1.2x	**68.1**	**88.1**	61
SYQ (Faraone et al., 2018)	2	8	–	67.7	87.8	–
Ours (ternary)	2	8	1.2x	**68.1**	**88.0**	61
TWN (Li & Liu, 2016)	2	32	–	61.8	84.2	–
INQ (Zhou et al., 2017)	2	32	–	66.0	87.1	–
TTQ (Zhu et al., 2016)	2	32	1.0x	66.6	87.2	–
LQ-Net (Zhang et al., 2018)	2	32	1.4x	68.0	88.0	–
Ours (ternary)	2	32	1.2x	**68.2**	**88.0**	61
Ours (quaternary-)	2	32	1.6x	**69.2**	**88.7**	57
LQ-Net (Zhang et al., 2018)	3	3	3.7x	68.2	87.9	–
LQ-Net (Zhang et al., 2018)	3	32	1.7x	69.3	88.8	–
INQ (Zhou et al., 2017)	3	32	–	68.1	88.4	–
Ours (quinary)	3	32	2.0x	**69.7**	**89.0**	53
LQ-Net (Zhang et al., 2018)	4	4	–	69.3	88.8	–
UNIQ (Baskin et al., 2018)	4	8	–	67.0	–	–
INQ (Zhou et al., 2017)	4	32	–	68.9	89.0	–
LQ-Net (Zhang et al., 2018)	4	32	–	70.0	89.1	–
UNIQ (Baskin et al., 2018)	5	8	–	68.0	–	–
INQ (Zhou et al., 2017)	5	32	–	69.0	89.1	–
Pruning (Liu et al., 2018)	32	32	–	66.5	87.3	40
Winograd-ReLU (Liu et al., 2018)	32	32	–	66.6	87.4	65
Inception-BN	bits	bits		%	%	%
TTQ	2	32	1.0x	67.0	87.3	58
Ours (ternary)	2	32	1.0x	**69.6**	**89.1**	58
Ours (quaternary-)	2	32	1.1x	**70.8**	**89.8**	52
Ours (quinary)	3	32	1.1x	**71.7**	**90.3**	54

architectures on real-world tasks, including ResNet-18 and ImageNet, show the effectiveness of our approach.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `http://arxiv.org/abs/1603.04467`.

Chaim Baskin, Eli Schwartz, Evgenii Zheltonozhskii, Natan Liss, Raja Giryes, Alexander M. Bronstein, and Avi Mendelson. UNIQ: uniform noise injection for the quantization of neural networks. *CoRR*, abs/1804.10969, 2018. URL `http://arxiv.org/abs/1804.10969`.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. *CoRR*, abs/1702.00953, 2017. URL http://arxiv.org/abs/1702.00953.

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL http://arxiv.org/abs/1602.02830.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015. URL http://arxiv.org/abs/1511.00363.

Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip H.W. Leong. Syq: Learning symmetric quantization for efficient deep neural networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015a. URL http://arxiv.org/abs/1510.00149.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015b. URL http://arxiv.org/abs/1506.02626.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, Feb 2014. doi: 10.1109/ISSCC.2014.6757323.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Fengfu Li and Bin Liu. Ternary weight networks. *CoRR*, abs/1605.04711, 2016. URL http://arxiv.org/abs/1605.04711.

Xingyu Liu, Jeff Pool, Song Han, and William J. Dally. Efficient sparse-winograd convolutional neural networks. *CoRR*, abs/1802.06367, 2018. URL http://arxiv.org/abs/1802.06367.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, March 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016. URL http://arxiv.org/abs/1603.05279.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Nikolaj V. Smirnov and Igor V. Dunin-Barkovskiĭ. *Mathematische Statistik in der Technik:*. Mathematik für Naturwissenschaften und Technik. Deutscher Verlag der Wissenschaften, 1963.

Yuxin Wu et al. Tensorpack. https://github.com/tensorpack/, 2016.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *European Conference on Computer Vision (ECCV)*, 2018.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044, 2017. URL http://arxiv.org/abs/1702.03044.

Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016. URL http://arxiv.org/abs/1606.06160.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. *CoRR*, abs/1612.01064, 2016. URL http://arxiv.org/abs/1612.01064.