COMPRESSING DEEP NEURAL NETWORKS WITH LEARNABLE REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider learning and compressing deep neural networks (DNNs) that consist of low-precision weights and activations for efficient inference of fixed-point operations. In training low-precision DNNs, gradient descent in the backward pass is performed with high-precision weights while quantized low-precision weights and activations are used in the forward pass for computing the loss function. Thus, the gradient descent becomes suboptimal, and accuracy loss follows. In order to reduce the mismatch in the forward and backward passes, we utilize mean squared quantization error (MSQE) regularization. In particular, we propose using a learnable regularization coefficient with the MSQE regularizer to reinforce the convergence of high-precision weights to their quantized values. Furthermore, we investigate how partial L2 regularization can be employed for weight pruning in a similar manner. Finally, combining weight pruning, quantization, and entropy coding, we establish a low-precision DNN compression pipeline. In our experiments, the proposed method produces low-precision MobileNet and ShuffleNet models on ImageNet classification with the state-of-the-art compression ratios. Moreover, we examine our method for image super resolution DNNs to produce low-precision models at negligible performance loss.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved performance breakthroughs in many of computer vision tasks (LeCun et al., 2015). The revolutionary progress of deep learning comes with over-parametrized multi-layer network architectures, and nowadays millions or tens of millions parameters in more than one hundred layers are not exceptional anymore. Network compression for efficient inference is of great interest for deployment of large-size DNNs on resource-limited platforms such as battery-powered mobile devices (Sze et al., 2017; Cheng et al., 2018). In such resource-constrained hardwares, not only memory and power are limited but also basic floating-point arithmetic operations are in some cases not supported. Hence, it is preferred and sometimes necessary to deliver compressed DNNs of low-precision fixed-point weights and activations (feature maps).

In this paper, we propose a network compression scheme that produces low-precision DNNs through learning with regularization. In particular, we let the regularization coefficient learnable, instead of treating it as a fixed hyper-parameter, to make a smooth and efficient transition of a high-precision model into a sparse quantized model. The proposed compression pipeline is summarized in Figure 1.

- For weight pruning, we utilize partial L2 regularization to make a portion of small-value weights tend to zero so we can safely prune them after training at negligible accuracy loss.
- For weight quantization, we regularize (unpruned) weights with another regularization term of the mean squared quantization error (MSQE). In this stage, we also quantize the activations (feature maps) of each layer to mimic low-precision operations at inference time. The quantization bin sizes for weights and activations are optimized to minimize their MSQEs in each layer.
- The pruned and quantized model is converted into a low-precision model and its low-precision weights are further compressed in size with lossless entropy coding such as Huffman coding and universal source coding algorithms (e.g., see Cover & Thomas, 2012, Section 11.3) for memory-efficient deployment.

It is difficult to train low-precision DNNs with standard gradient descent since the learning rate is typically set to be a small floating-point value but low-precision weights cannot be adjusted in fine

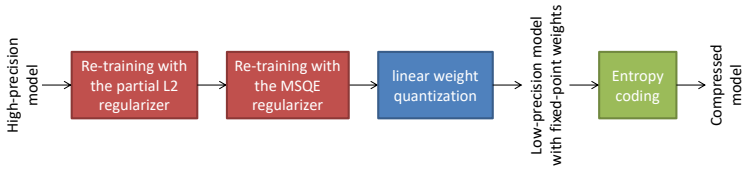


Figure 1: Our low-precision DNN compression pipeline. We utilize partial L2 regularization and MSQE regularization to transform a pre-trained high-precision model into a sparse low-precision model with fixed-point weights and activations. The low-precision weights are further compressed in size with lossless entropy source coding.

resolution. To enable training low-precision DNNs, a series of papers on binary neural networks suggests utilizing high-precision shadow weights to accumulate the negatives of the gradients in fine resolution, while the gradients are obtained from the network loss function calculated with binarized (or quantized) weights (Courbariaux et al., 2015; Lin et al., 2016; Hubara et al., 2016). That is, high-precision weights are quantized in the forward pass, but the quantization function is replaced with the linear function of slope 1 in the backward pass for gradient descent. This approximate gradient descent algorithm is further refined in subsequent works (Rastegari et al., 2016; Zhou et al., 2016; Zhu et al., 2017; Cai et al., 2017; Hou et al., 2017; Hou & Kwok, 2018; Gysel et al., 2018; Zhou et al., 2018).

The mismatch in the forward and backward passes results in sub-optimal gradient descent that causes accuracy loss. The mismatch is more problematic for the models using lower-precision weights and activations, since the quantization error is more significant. There have been some attempts to reduce this mismatch by introducing better backward pass approximation, e.g., using clipped ReLU and log-tailed ReLU instead of the linear function (e.g., see Cai et al., 2017). Our approach is different from these efforts. We use regularization to steer high-precision weights to converge to their quantized values so that the mismatch between high-precision weights and quantized weights becomes smaller instead of enhancing the backward pass approximation.

We reduce the mismatch between high-precision weights and quantized weights with MSQE regularization. In particular, we propose making the regularization coefficient learnable. Using learnable regularization, high-precision weights are reinforced to converge to their quantized values gradually in training. We empirically show that our learnable regularization yields more accurate low-precision models than the conventional regularization with a fixed regularization coefficient. Our scheme is different from the loss-aware weight quantization in Hou et al. (2017); Hou & Kwok (2018), where optimization solutions using the proximal Newton algorithm are presented to minimize the loss function under the constraints of low-precision weights, which is however impractical for large-size networks due to the prohibitive computational cost to estimate the Hessian matrix of the loss function. No regularization is considered in Hou et al. (2017); Hou & Kwok (2018).

Weight pruning curtails redundant weights completely from DNNs so that one can skip computations for pruned ones. Some of successful pruning algorithms can be found in Han et al. (2015); Lebedev & Lempitsky (2016); Wen et al. (2016); Guo et al. (2016); Lin et al. (2017). In this paper, we discuss how partial L2 regularization can be used for weight pruning. Combining weight pruning, quantization, and entropy coding, as shown in Figure 1, we achieve the state-of-the-art compression results for low-precision MobileNet (Howard et al., 2017) and ShuffleNet (Zhang et al., 2018) on ImageNet classification.

Weight sharing is another network compression scheme studied in Han et al. (2016); Choi et al. (2017); Ullrich et al. (2017); Molchanov et al. (2017); Agustsson et al. (2017); Louizos et al. (2017); Choi et al. (2018); Tung & Mori (2018). It reduces the number of distinct weight values in DNNs by quantization. In contrast to low-precision weights from linear quantization, weight sharing allows non-linear quantization, where quantization output levels do not have to be evenly spaced. Hence, quantized weights from weight sharing are represented in high precision, implying that high-precision arithmetic operations are still needed in inference, although the quantized weights can be compressed in size by lossless source coding.

We finally note that reinforcement learning has been proposed as a promising methodology to search for quantized and/or compressed models that satisfy certain latency, energy, and/or model size requirements, given hardware specifications to deploy the models (He et al., 2018; Wang et al., 2019).

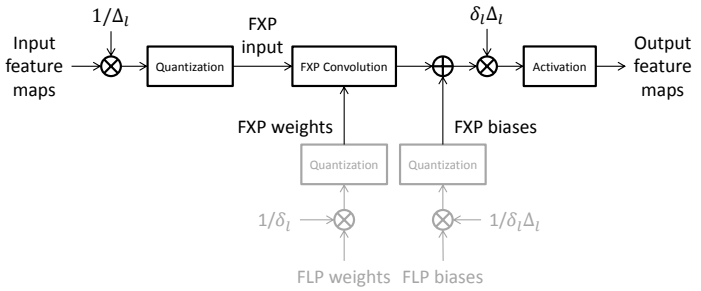


Figure 2: Low-precision convolutional layer using fixed-point (FXP) convolution and bias addition.

2 LOW-PRECISION DNN MODEL

We consider low-precision DNNs that are capable of efficient processing in the inference stage by using fixed-point arithmetic operations. In particular, we focus on the fixed-point implementation of convolutional and fully-connected layers, since they are the dominant parts of computational costs and memory requirements in DNNs (see Sze et al., 2017, Table II).

The major bottleneck of efficient DNN processing is known to be in memory accesses (Sze et al., 2017, Section V-B). Horowitz provides rough energy costs of various arithmetic and memory access operations for 45 nm technology (see Horowitz, 2014, Figure 1.1.9), where we can find that memory accesses typically consume more energy than arithmetic operations, and the memory access cost increases with the read size. Hence, for example, deploying binary models, instead of 32-bit models, it is expected to reduce energy consumption by $32\times$ at least, due to 32 times fewer memory accesses.

Low-precision weights and activations basically stem from linear quantization (e.g., see Gersho & Gray, 2012, Section 5.4), where quantization bin boundaries are uniformly spaced and quantization output levels are the midpoints of bin intervals. Quantized weights and activations are represented by fixed-point numbers of small bit-width. Scaling factors (i.e., quantization bin sizes) are defined in each layer for fixed-point weights and activations, respectively, to alter their dynamic ranges.

Figure 2 shows the fixed-point design of a general convolutional layer consisting of convolution, bias addition and non-linear activation. Fixed-point weights and input feature maps are given with common scaling factors δ_l and Δ_l, respectively, where l is the layer index. Then, the convolution operation can be implemented by fixed-point multipliers and accumulators. Biases are added, if present, after the convolution, and then the output is scaled properly by the product of the scaling factors for weights and input feature maps, i.e., $\delta_l\Delta_l$, as shown in the figure. Here, the scaling factor for the biases is specially set to be $\delta_l\Delta_l$ so that fixed-point bias addition can be done easily without another scaling. Then, a non-linear activation function follows. Finally, the output activations are fed into the next layer as the input.

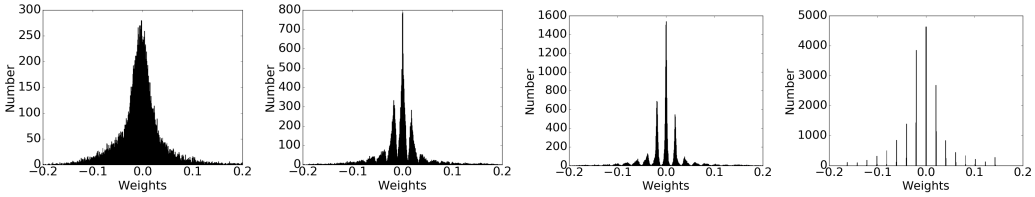
Using rectified linear unit (ReLU) activation, two scaling operations across two layers, i.e., scaling operations by $\delta_l\Delta_l$ and $1/\Delta_{l+1}$, can be combined into one scaling operation by $\delta_l\Delta_l/\Delta_{l+1}$ before (or after) ReLU activation. Furthermore, if the scaling factors are power-of-two numbers, then one can even implement scaling by bit-shift. Similarly, low-precision fully-connected layers can be implemented by replacing convolution with matrix multiplication in the figure.

3 LEARNABLE REGULARIZATION FOR LOW-PRECISION DNNS

In this section, we present the regularizers that are utilized to learn pruned and quantized DNNs of low-precision weights and activations. We first define the quantization function. Given the number of bits, i.e., bit-width n, the quantization function yields

$$Q_n(x;\delta) = \begin{cases} \delta\,\mathrm{clip}_n(\mathrm{round}(x/\delta)), & n \geq 2, \\ \delta\,\mathrm{sign}(x), & n = 1, \end{cases} \tag{1}$$

where x is the input and δ is the scaling factor; $\mathrm{round}(x) = \mathrm{sign}(x)\lfloor |x| + 0.5 \rfloor$ and $\mathrm{clip}_n(x) = \min(\max(x, -2^{n-1}), 2^{n-1} - 1)$, where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x. For



(a) Iterations=$10k$ (b) Iterations=$21k$ (c) Iterations=$23k$ (d) Iterations=$30k$

Figure 3: Weight histogram snapshots of the MNIST LeNet-5 second convolutional layer captured at different training batch iteration numbers while a pre-trained model is quantized to have 4-bit weights and activations with the proposed regularization method.

ReLU activation, the ReLU output is always non-negative, and thus we use the unsigned quantization function given by

$$Q_n^+(x;\delta) = \delta \,\mathrm{clip}_n^+(\mathrm{round}(x/\delta)), \tag{2}$$

for $n \geq 1$, where $\mathrm{clip}_n^+(x) = \min(\max(x, 0), 2^n - 1)$.

3.1 Regularization for weight quantization

Consider a general non-linear DNN model consisting of L layers. Let $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_L$ be the sets of high-precision weights in layers 1 to L, respectively. For notational simplicity, we let $A_1^L = A_1, A_2, \ldots, A_L$ for any symbol A. We define the MSQE regularizer for weights of all L layers as

$$R_n(\mathcal{W}_1^L; \delta_1^L) = \frac{1}{N} \sum_{l=1}^{L} \sum_{w \in \mathcal{W}_l} |w - Q_n(w; \delta_l)|^2, \quad N = \sum_{l=1}^{L} |\mathcal{W}_l|, \tag{3}$$

where n is the bit-width for quantized weights, δ_l is the scaling factor (i.e., quantization bin size) for quantized weights, and $|\mathcal{W}_l|$ is the number of weights in layer l. We assumed that bit-width n is the same for all layers, just for notational simplicity, but it can be easily extended to more general cases such that each layer has a different bit-width.

Including the MSQE regularizer in (3), the cost function to optimize in training is given by

$$C_n(\mathcal{X}; \mathcal{W}_1^L, \delta_1^L) = E(\mathcal{X}; Q_n(\mathcal{W}_1^L; \delta_l^L)) + \lambda R_n(\mathcal{W}_1^L; \delta_1^L), \tag{4}$$

for $\lambda > 0$, where, with a slight abuse of notation, $Q_n(\mathcal{W}_1^L; \delta_l^L)$ denotes the set of quantized weights of all L layers, $E(\mathcal{X}; Q_n(\mathcal{W}_1^L))$ is the target loss function evaluated on the training dataset $\mathcal{X}$ using the quantized weights, and λ is the regularization coefficient. We set the scaling factors δ_1^L to be learnable parameters and optimize them along with weights $\mathcal{W}_1^L$.

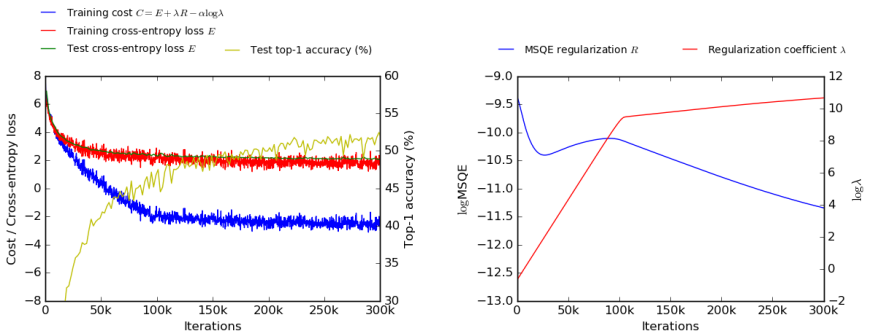
Remark 1. We clarify that we use high-precision weights in the backward pass for gradient descent by replacing approximately the quantization function Q_n with the linear function of slope 1. In the forward pass, we use quantized weights and activations, and the main target network loss function E is also calculated with the quantized weights and activations to mimic the low-precision inference-stage loss. Hence, the final trained models are low-precision models, which can be operated on low-precision fixed-point hardwares in inference with no accuracy loss. Note that our method still has the gradient mismatch problem, similar to the existing approaches (see Section 1). However, by adding the MSQE regularizer, we encourage high-precision weights to converge to their quantized values so that we reduce the mismatch.

Learnable regularization coefficient. The regularization coefficient λ in (4) is a hyper-parameter that controls the trade-off between the loss and the regularization. It is conventionally fixed ahead of training. However, searching for a good hyper-parameter value is usually time-consuming. Hence, we propose the learnable regularization coefficient, i.e., we let the regularization coefficient be another learnable parameter.

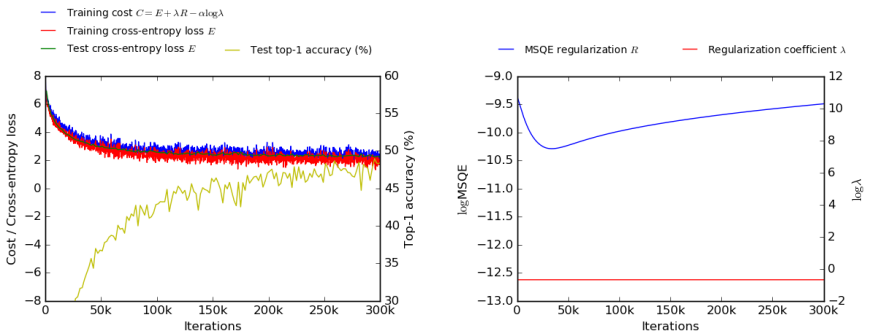
We start training with a small initial value for λ, i.e., with little regularization. However, we promote the increase of λ in training by adding a penalty term for a small regularization coefficient, which is $-\alpha \log \lambda$ for $\lambda, \alpha > 0$, in the cost function (see (5)). The increasing coefficient λ reinforces the convergence of high-precision weights to their quantized values for reducing the MSQE. It consequently alleviates the gradient mismatch problem that we mentioned earlier (see Remark 1).

Table 1: Comparison of the weight regularization methods with learnable and fixed regularization coefficients for ResNet-18 on ImageNet classification.

Weights	Activations	Top-1 / Top-5 accuracy (%)			
		Learnable λ	Fixed $\lambda = 0.05$	Fixed $\lambda = 0.5$	Fixed $\lambda = 5$
32-bit FLP	32-bit FLP	68.1 / 88.4			
1-bit FXP	8-bit FXP	61.3 / 83.7	60.0 / 83.1	60.0 / 83.0	57.9 / 81.6
	4-bit FXP	60.2 / 83.2	58.1 / 81.5	57.4 / 81.1	58.6 / 82.2
	2-bit FXP	55.6 / 79.6	53.5 / 78.2	52.9 / 77.8	53.1 / 78.1
	1-bit FXP	38.9 / 65.4	37.0 / 63.4	36.5 / 63.1	37.0 / 63.1



(a) Learnable λ



(b) Fixed $\lambda = 0.5$

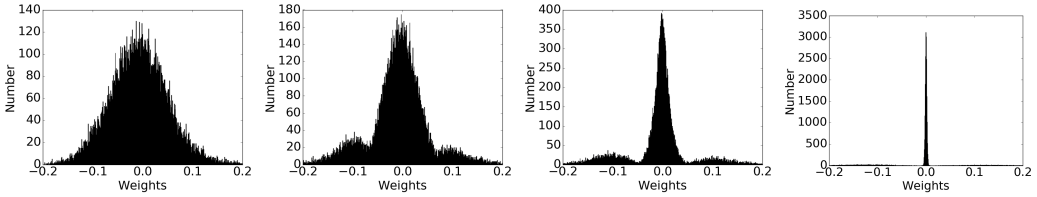
Figure 4: ResNet-18 model training convergence curves for binary weights and 2-bit activations. We compare the convergence curves with learnable and fixed regularization coefficients.

The cost function in (4) is altered into

$$C_n(\mathcal{X}; \mathcal{W}_1^L, \delta_1^L, \lambda) = E(\mathcal{X}; Q_n(\mathcal{W}_1^L; \delta_l^L)) + \lambda R_n(\mathcal{W}_1^L; \delta_1^L) - \alpha \log \lambda, \tag{5}$$

where we introduce a new hyper-parameter α, while making the regularization coefficient learnable. We note that the trade-off between the loss and the regularization is now actually controlled by the new parameter α instead of λ, i.e., the larger the value of α, eventually the more the regularization. This transfer is however beneficial since the new parameter α does not directly impact either the loss or the regularization, which leads to a smooth transition of high-precision weights to their quantized values. For gradient descent, we need the gradients of (5) with respect to weights, scaling factors and the regularization coefficient, respectively, which are provided in Appendix A.1.

Learnable versus fixed regularization coefficients. In Table 1, we compare the performance of quantized ResNet-18 (He et al., 2016) models for ImageNet ILSVRC 2012 dataset (Russakovsky et al., 2015) when we use learnable and fixed regularization coefficients, respectively. Observe that the proposed learnable regularization method outperforms the conventional regularization method with a fixed coefficient in various low-precision settings.



(a) Iterations=$4k$ (b) Iterations=$6k$ (c) Iterations=$8k$ (d) Iterations=$10k$

Figure 5: Weight histogram snapshots of the MNIST LeNet-5 at different training batch iteration numbers when trained from scratch with the partial L2 regularizer for 90% sparsity ($r = 90$).

In Figure 4, we compare the convergence curves when learnable and fixed regularization coefficients are used, respectively. Using a learnable regularization coefficient, the MSQE regularization term decreases (although there is a bump in the middle) while λ increases in training. However, using a fixed regularization coefficient, the MSQE regularization term saturates and even increases after some point as training goes on, which implies that the mismatch of the forward and backward passes is not resolved. The unresolved mismatch eventually turns into accuracy loss, as shown in the figure.

Evolution of weight histogram. Figure 3 presents an example of how high-precision weights are gradually quantized by our regularization scheme. We plotted weight histogram snapshots captured at the second convolutional layer of the MNIST LeNet-5 model[1] while a pre-trained model is quantized to a 4-bit fixed-point model. The histograms in the figure from the left to the right correspond to $10k$, $21k$, $23k$, and $30k$ batch iterations in training, respectively. Observe that the weight distribution gradually converges to the sum of uniformly spaced delta functions and all high-precision weights converge to quantized values completely in the end.

Comparison to soft weight sharing. In soft weight sharing (Nowlan & Hinton, 1992; Ullrich et al., 2017), a Gaussian mixture prior is assumed, and the model is regularized to form groups of weights that have similar values around the Gaussian component centers (e.g., see Bishop, 2006, Section 5.5.7). The learnable regularization coefficient can be related to the learnable variance in the Gaussian mixture prior. However, our weight regularization method is different from soft weight sharing since we consider linear quantization and optimize quantization bin sizes, instead of optimizing individual Gaussian component centers for non-linear quantization. We employ the simple MSQE regularization term for quantization, so that it is applicable to large-size DNNs. Note that soft weight sharing yields the regularization term of the logarithm of the summation of exponential functions, which is sometimes too complex to compute for large-size DNNs. In our method, the additional computational complexity for MSQE regularization is not expensive. It only scales in the order of $O(N)$, where N is the number of weights. Hence, the proposed scheme is easily applicable to the state-of-the-art DNNs with millions or tens of millions weights.

3.2 Quantization of activations

We quantize the output activation (feature map) x of layer l for $1 \leq l \leq L$ and yield $Q_m^+(x; \Delta_l)$, where Q_m^+ is the quantization function in (2) for bit-width m and Δ_l is the learnable scaling factor for quantized activations of layer l.[2] Similar to (3), we assumed that activation bit-width m is the same for all layers, but this constraint can be easily relaxed to cover the cases where each layer has a different bit-width. We assumed ReLU activation and used the unsigned quantization function Q_m^+ while we can replace Q_m^+ with Q_m in case of general non-linear activation.

We optimize Δ_l by minimizing the MSQE for activations of layer l, i.e., we minimize

$$S_m(\mathcal{A}_l; \Delta_l) = \frac{1}{|\mathcal{A}_l|} \sum_{x \in \mathcal{A}_l} |x - Q_m^+(x; \Delta_l)|^2, \tag{6}$$

[1] https://github.com/BVLC/caffe/tree/master/examples/mnist

[2] We note that Δ_l is the scaling factor for activations of layer l whereas it denotes the scaling factor for input feature maps of layer l in Section 2 (see Figure 2). This is just one index shift in the notation, since the output of layer l is the input to layer $l + 1$. We adopt this change just for notational simplicity.

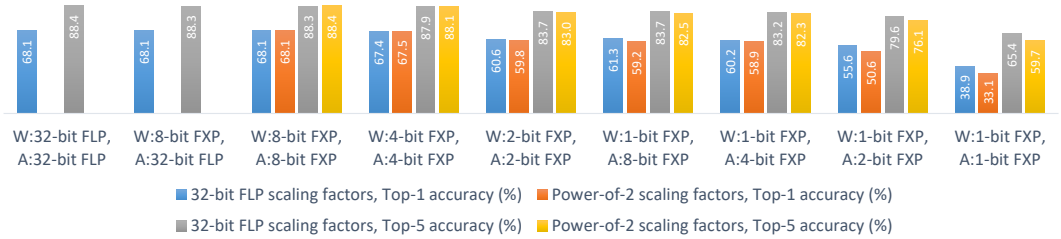


Figure 6: Ablation study of ResNet-18 quantization on ImageNet classification. We use "W: Weight precision" and "A: Activation precision" to denote weight and activation precisions, respectively. FLP and FXP stands for floating-point and fixed-point formats, respectively.

where $\mathcal{A}_l$ is the set of activations of layer l for $1 \leq l \leq L$. In the backward pass, we first perform gradient descent for weights and their scaling factors using the loss function in (5), and then we update Δ_l with gradient descent using (6). We do not utilize (6) in gradient descent for weights.

Backpropagation through quantized activations: Backpropagation is not feasible through quantized activations analytically since the gradient is zero almost everywhere. For backpropagation through the quantization function, we adopt the straight-through estimator (Bengio et al., 2013). In particular, we pass the gradient through the quantization function when the input is within the clipping boundary. If the input is outside the clipping boundary, we pass zero.

3.3 Regularization for weight pruning

For weight pruning, we propose using partial L2 regularization. In particular, given a target pruning ratio r, we find the r-th percentile of weight magnitude values. Assuming that we prune the weights below this r-th percentile value in magnitude, we define a L2 regularizer only for them as follows:

$$P_r(\mathcal{W}_1^L) = \frac{1}{N} \sum_{l=1}^{L} \sum_{w \in \mathcal{W}_l} |w|^2 1_{|w|<\theta(r)},$$

where $\theta(r)$ is the r-th percentile of weight magnitude values, which is the threshold for pruning. Adopting the learnable regularization coefficient as in (5), we have

$$C_r(\mathcal{X}; \mathcal{W}_1^L, \lambda) = E(\mathcal{X}; \mathcal{W}_1^L) + \lambda P_r(\mathcal{W}_1^L) - \alpha \log \lambda. \tag{7}$$

The partial L2 regularizer encourages the weights below the threshold to move towards zero, while the other unregularized weights are updated to minimize the loss due to pruning. The threshold $\theta(r)$ is also updated at every iteration of training based on the instant weight distribution. We note that the threshold $\theta(r)$ decreases as training goes on since the regularized weights gradually converge to zero (see Figure 5). After finishing the regularized training, we finally have a set of weights clustered very near zero. The loss due to pruning these small-value weights is negligible.

4 Experiments

We evaluate the proposed low-precision DNN compression for ImageNet classification and image super resolution. Image super resolution is included in our experiments as a regression problem since its accuracy is more sensitive to quantization than classification. Note that Tensorflow Lite[3] already supports a very efficient 8-bit weight and activation quantization tool for network development on mobile platforms. Thus, our experimental results focus on more extreme cases of quantization using less than 8 bits, where a more sophisticated algorithm is needed for smaller loss. The detailed experimental settings and some of the experimental results are provided in Appendix B due to the space constraints. We use FLP and FXP to denote the floating-point and fixed-point formats, respectively.

4.1 Experimental results

ResNet-18 quantization on ImageNet classification. Figure 6 presents the accuracy of the low-precision ResNet-18 (He et al., 2016) models obtained from our quantization method. For ablation

[3] https://www.tensorflow.org/lite

Table 2: Low-precision MobileNet and ShuffleNet compression results for ImageNet classification. For ablation study, we compare pruning-only results and pruning+quantization results with various low-precision setting. We also show the compression results with and without entropy coding, where we used bzip2 as a specific entropy coding scheme.

Method	Weights	Activations	MobileNet v1		ShuffleNet	
			Top-1 / Top-5 accuracy (%)	Compression ratio with / without bzip2	Top-1 / Top-5 accuracy (%)	Compression ratio with / without bzip2
Pre-trained model	32-bit FLP	32-bit FLP	70.9 / 89.9	-	65.4 / 86.4	-
Ours: pruning (50%)	32-bit FLP	32-bit FLP	70.2 / 89.7	2.01 / 1.00	65.3 / 86.4	1.99 / 1.00
pruning (55%)	32-bit FLP	32-bit FLP	70.0 / 89.5	2.22 / 1.00	64.7 / 86.0	2.20 / 1.00
pruning (60%)	32-bit FLP	32-bit FLP	69.5 / 89.3	2.49 / 1.00	63.6 / 85.5	2.45 / 1.00
Ours: pruning (50%) + quantization	8-bit FXP	8-bit FXP	70.8 / 90.1	4.83 / 4.00	65.8 / 86.7	4.99 / 4.00
	6-bit FXP	6-bit FXP	70.5 / 89.9	6.11 / 5.33	65.7 / 86.7	5.81 / 5.33
	5-bit FXP	5-bit FXP	69.7 / 89.3	7.13 / 6.40	64.0 / 85.6	6.78 / 6.40
	4-bit FXP	4-bit FXP	66.9 / 87.7	9.87 / 8.00	59.5 / 82.6	9.59 / 8.00
	6-bit FXP	8-bit FXP	70.6 / 90.0	6.11 / 5.33	66.3 / 87.1	5.81 / 5.33
	5-bit FXP		70.3 / 89.7	7.13 / 6.40	65.8 / 86.7	6.79 / 6.40
	4-bit FXP		69.7 / 89.2	8.65 / 8.00	64.8 / 86.2	8.26 / 8.00
	6-bit FXP	32-bit FLP	70.7 / 90.0	6.12 / 5.33	66.3 / 87.1	5.81 / 5.33
	5-bit FXP		70.4 / 89.8	7.13 / 6.40	65.8 / 86.9	6.78 / 6.40
	4-bit FXP		69.3 / 89.0	10.01 / 8.00	64.1 / 85.8	9.71 / 8.00
Tensorflow 8-bit model*	8-bit FXP	8-bit FXP	70.1 / 88.9	N/A / 4.00	N/A / N/A	N/A

* https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md

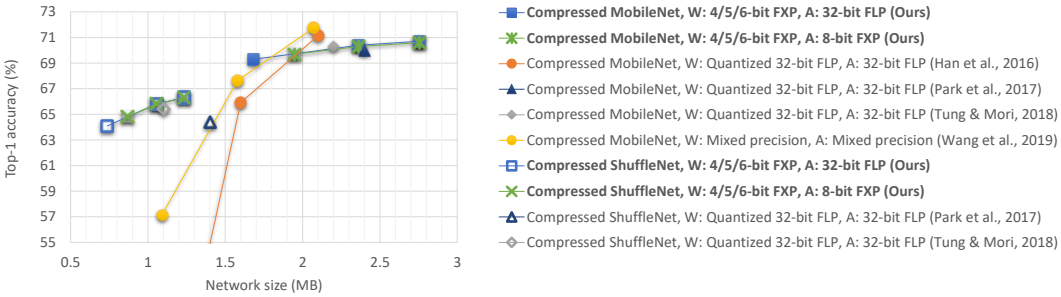


Figure 7: Comparison of our low-precision MobileNet and ShuffleNet compression results to the ones of the state-of-the-art network compression methods on ImageNet classification. We use "W: Weight precision" and "A: Activation precision" to denote weight and activation precisions used in the compressed models, respectively.

study, we compare weight and activation quantization for various low-precision settings. The loss due to weight quantization is relatively less than the loss due to activation quantization, which is consistent with the results from DoReFa-Net (Zhou et al., 2016). We compare our method to DoReFa-Net for AlexNet (Krizhevsky et al., 2012) quantization, which can be found in Appendix B.2. We also compare the low-precision models obtained with and without the power-of-two scaling constraint. In fixed-point computations (see Figure 2), it is more appealing for the scaling factors (i.e., quantization bin sizes) to be powers of two so they can be implemented by simple bit-shift, rather than with scalar multiplication. For power-of-two scaling factors, we perform rounding of scaling factors into their closest power-of-two integers in the forward pass, while the rounding function is replaced with the linear function of slope 1 in the backward pass. We observe small performance degradation due to the constraint of power-of-two scaling factors in our experiments.

MobileNet/ShuffleNet compression on ImageNet classification. We mainly evaluate our method to obtain compressed low-precision MobileNet (Howard et al., 2017) and ShuffleNet (Zhang et al., 2018) models for ImageNet classification. For MobileNet and ShuffleNet compression, we prune 50% weights from their pre-trained models as described in Section 3.3 so that the accuracy loss due to pruning is marginal. Then, we employ our weight and activation quantization method. After converting into sparse low-precision models, universal source coding with bzip2 (Seward, 1998) follows to compress the fixed-point low-precision weights.

Table 3: CT-SRCNN (9-layer) quantization results for upscaling factor 3.

Model	Method	Weights	Activations	Set-14 PSNR (dB)	Set-14 SSIM	PSNR (dB) loss	SSIM loss
CT-SRCNN 9-layer	Pretrained model	32-bit FLP	32-bit FLP	29.71	0.8300	-	-
	Ours	8-bit FXP	8-bit FXP	29.67	0.8288	0.04	0.0012
		4-bit FXP		29.63	0.8285	0.08	0.0015
		2-bit FXP		29.37	0.8236	0.34	0.0064
		1-bit FXP		29.20	0.8193	0.51	0.0107
	Gysel et al. (2018)*	8-bit FXP	8-bit FXP	29.05	0.8065	0.74	0.0234
Bicubic	-	-	-	27.54	0.7742	-	-

* from our experiments using their code at `https://github.com/pmgysel/caffe`.

In Figure 7, we compare the compression ratios of our scheme and the existing network compression methods in Han et al. (2016); Park et al. (2017); Tung & Mori (2018); Wang et al. (2019). Our low-precision network compression scheme shows comparable compression ratios to the state-of-the-art weight compression schemes. We emphasize that our compression scheme produces low-precision models of fixed-point weights and activations that support efficient inference of fixed-point operations, while the previous compression schemes, except the one from Wang et al. (2019), produces quantized weights that are still floating-point numbers and thus floating-point operations are necessary to achieve the presented accuracy of them. The hardware-aware automated quantization in Wang et al. (2019) achieved impressive compression results by searching for a quantized model with "mixed" precisions for different layers with reinforcement learning, but not all hardwares support mixed precision arithmetic operations. Our method used a fixed precision for all layers assuming more limited hardware capability.

Image super resolution. The image super resolution problem is to synthesize a high-resolution image from a low-resolution one. The DNN output is the high-resolution image corresponding to the input low-resolution image, and thus the loss due to quantization is more prominent. We evaluate the proposed method on SRCNN (Dong et al., 2016) and cascade-trained SRCNN (CT-SRCNN) (Ren et al., 2018) for image super resolution. The objective image quality metric measured by the peak signal-to-noise ratio (PSNR) and the perceptual score measured by the structural similarity index (SSIM) (Wang et al., 2004) are compared for Set-14 image dataset (Zeyde et al., 2010) in Table 3 for 9-layer CT-SRCNN. Additional experimental results for 3-layer SRCNN and 5-layer CT-SRCNN can be found in Appendix B.2. Observe that our method successfully yields low-precision models of 8-bit weights and activations at negligible loss, and they are better than the results that we obtain with one of the previous works, Ristretto (Gysel et al., 2018). It is interesting to see that the PSNR loss of using binary weights and 8-bit activations is 0.5 dB only.

5 Conclusion

In this paper, we proposed a method to quantize deep neural networks (DNNs) by regularization to produce low-precision DNNs for efficient fixed-point inference. The proposed scheme alleviates the mismatch problem in the forward and backward passes of low-precision network training by using MSQE regularization. Moreover, we proposed a novel learnable regularization coefficient to reinforce the convergence of high-precision weights to their quantized values when using MSQE regularization. We also discussed how a similar regularization technique can be employed for weight pruning with partial L2 regularization.

We showed by experiments that the proposed quantization algorithm successfully produces low-precision DNNs of binary weights for classification problems, such as ImageNet classification, as well as for regression and image synthesis problems, such as image super resolution. For MobileNet and ShuffleNet compression, we obtained sparse (50% weights are pruned) low-precision models of 5-bit weights and 8-bit activations with compression ratios of 7.13 and 6.79, respectively, at marginal accuracy loss. For image super resolution, we only lost 0.04 dB PSNR when using 8-bit weights and activations, instead of 32-bit floating-point numbers.

REFERENCES

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pp. 1141–1151, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave Gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5918–5926, 2017.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35 (1):126–136, 2018.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. In *International Conference on Learning Representations*, 2017.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Universal deep neural network compression. In *NeurIPS Workshop on Compact Deep Neural Network Representation with Industrial Applications (CDNNRIA)*, 2018.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pp. 3123–3131, 2015.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2): 295–307, 2016.

Allen Gersho and Robert M Gray. *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media, 2012.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient DNNs. In *Advances In Neural Information Processing Systems*, pp. 1379–1387, 2016.

Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, (99):1–6, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference*, pp. 10–14, 2014.

Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. In *International Conference on Learning Representations*, 2018.

Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. In *International Conference on Learning Representations*, 2017.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, pp. 4107–4115, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2554–2564, 2016.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pp. 2178–2188, 2017.

Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. In *International Conference on Learning Representations*, 2016.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3290–3300, 2017.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pp. 2498–2507, 2017.

Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992.

Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7197–7205, 2017.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Haoyu Ren, Mostafa El-Khamy, and Jungwon Lee. CT-SRCNN: Cascade trained and trimmed deep convolutional neural networks for image super resolution. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Julian Seward. bzip2, 1998. URL `www.bzip.org`.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

Frederick Tung and Greg Mori. Deep neural network compression by in-parallel pruning-quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations*, 2017.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pp. 711–730. Springer, 2010.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

Aojun Zhou, Anbang Yao, Kuan Wang, and Yurong Chen. Explicit loss-error-aware quantization for low-bit deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9426–9435, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *International Conference on Learning Representations*, 2017.

Appendices

A GRADIENT DESCENT

A.1 GRADIENTS FOR WEIGHTS

The gradient of the cost function C_n in (5) with respect to weight w satisfies

$$\nabla_w C_n = \nabla_w E + \lambda \nabla_w R_n, \tag{8}$$

for weight w of layer l, $1 \leq l \leq L$. The first partial derivative in the right side of (8) can be obtained efficiently by the backpropagation algorithm. For backpropgation through the weight quantization function, we adopt the following approximation similar to straight-through estimator (Bengio et al., 2013):

$$\nabla_w Q_n(w; \delta_l) \triangleq \begin{cases} 1_{\frac{w}{\delta_l} \in [-2^{n-1} - \frac{1}{2}, 2^{n-1} - \frac{1}{2}]}, & n > 1, \\ 1_{\frac{w}{\delta_l} \in [-2, 2]}, & n = 1, \end{cases} \tag{9}$$

where $1_{\mathcal{E}}$ is an indication function such that it is one if $\mathcal{E}$ is true and zero otherwise. Namely, we pass the gradient through the quantization function when the weight is within the clipping boundary. To give some room for the weight to move around the boundary in stochastic gradient descent, we additionally allow some margin of $\delta_l/2$ for $n \geq 2$ and δ_l for $n = 1$. Outside the clipping boundary with some margin, we pass zero.

For weight w of layer l, $1 \leq l \leq L$, the partial derivative of the regularizer R_n satisfies

$$\nabla_w R_n = \frac{2}{N}(w - Q_n(w; \delta_l)), \tag{10}$$

almost everywhere except some non-differentiable points of w at quantization bin boundaries $\mathcal{U}_n(\delta_l)$ given by

$$\mathcal{U}_n(\delta_l) = \left\{ \frac{2i + 1 - 2^n}{2} \delta_l, i = 0, 1, \ldots, 2^n - 2 \right\}, \tag{11}$$

for $n > 1$ and $\mathcal{U}_1(\delta_l) = \{0\}$. If the weight is located at one of these boundaries, it actually makes no difference to update w to either direction of $w - \epsilon$ or $w + \epsilon$, in terms of its quantization error. Thus, we let

$$\nabla_w R_n \triangleq 0, \quad \text{if } w \in \mathcal{U}_n(\delta_l). \tag{12}$$

From (8)–(12), we finally have

$$\nabla_w C_n = \nabla_w E + \frac{2\lambda}{N}(w - Q_n(w; \delta_l)) 1_{w \notin \mathcal{U}_n(\delta_l)}. \tag{13}$$

Remark 2. If the weight is located at one of the bin boundaries, the weight gradient is solely determined by the network loss function derivative and thus the weight is updated towards the direction to minimize the network loss function. Otherwise, the regularization term impacts the gradient as well and encourages the weight to converge to the closest bin center as far as the loss function changes small. The regularization coefficient trades off these two contributions of the network loss function and the regularization term.

A.2 GRADIENT FOR THE REGULARIZATION COEFFICIENT

The gradient of the cost function for λ is given by

$$\nabla_\lambda C_n = R_n(\mathcal{W}_1^L; \delta_1^L) - \frac{\alpha}{\lambda}. \tag{14}$$

Observe that λ tends to α / R_n in gradient descent.

Remark 3. Recall that weights gradually tend to their closest quantization output levels to reduce the regularizer R_n (see Remark 2). As the regularizer R_n decreases, the regularization coefficient λ gets larger by gradient descent using (14). Then, a larger regularization coefficient further forces weights to move towards quantized values in the following update. In this manner, weights gradually converges to quantized values.

A.3 GRADIENTS FOR SCALING FACTORS

For scaling factor optimization, we approximately consider the MSQE regularization term only for simplicity. Using the chain rule, it follows that

$$\nabla_{\delta_l} C_n \approx \nabla_{\delta_l} R_n = -\frac{2\lambda}{N} \sum_{w \in \mathcal{W}_l} (w - Q_n(w; \delta_l)) \nabla_{\delta_l} Q_n(w; \delta_l), \tag{15}$$

for $1 \leq l \leq L$. Moreover, it can be shown that

$$\nabla_{\delta_l} Q_n(w; \delta_l) = r_n(w; \delta_l) \triangleq \begin{cases} \text{clip}_n(\text{round}(w/\delta_l)), & n > 1, \\ \text{sign}(w), & n = 1, \end{cases} \tag{16}$$

almost everywhere except some non-differentiable points of δ_l satisfying

$$\frac{w}{\delta_l} \in \left\{ \frac{2i + 1 - 2^n}{2}, i = 0, 1, \ldots, 2^n - 2 \right\}, \tag{17}$$

for $n > 1$. Similar to (12), we let

$$\nabla_{\delta_l} Q_n(w; \delta_l) \triangleq 0, \quad \text{if } w \in \mathcal{U}_n(\delta_l), \tag{18}$$

so that the scaling factor δ_l is not impacted by the weights at the bin boundaries. From (15)–(18), it follows that

$$\nabla_{\delta_l} C_n \approx -\frac{2\lambda}{N} \sum_{w \in \mathcal{W}_l} (w - Q_n(w; \delta_l)) r_n(w; \delta_l) 1_{w \notin \mathcal{U}_n(\delta_l)}.$$

Similarly, one can derive the gradients for activation scaling factors Δ_0^L, which we omit here.

B EXPERIMENTS

B.1 EXPERIMENTAL SETTINGS

Datasets. For ImageNet classification, we use the ImageNet ILSVRC 2012 dataset (Russakovsky et al., 2015). For image super resolution, we use the Open Images dataset[4], which is pre-processed as described in Ren et al. (2018).

Computing infrastructure. We use NVIDIA GeForce GTX TITAN X in our experiments. The proposed network pruning, quantization, and compression pipeline is implemented with Caffe[5].

Pre-trained models. The pre-trained models used in our ImageNet classification experiments are obtained from the following links.

AlexNet	`https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`
ResNet-18	`https://github.com/HolmesShuan/ResNet-18-Caffemodel-on-ImageNet`
MobileNet	`https://github.com/shicai/MobileNet-Caffe`
ShuffleNet	`https://github.com/msnqqer/ShuffleNet`

For image super resolution, we train (CT-)SRCNNs from scratch following Ren et al. (2018).

Hyper-parameters. We employ the Adam optimizer (Kingma & Ba, 2014). The learning rate is set to be 10^{-5} and we train 300k batches with the batch size of 256, 128, 32 and 64 for AlexNet, ResNet-18, MobileNet and ShuffleNet, respectively. Then, we decrease the learning rate to 10^{-6} and train $200k$ more batches. We use $\alpha = 0.5$. For the learnable regularization coefficient λ, we let $\lambda = e^{\omega}$ and learn ω instead in order to make λ always positive in training. The initial value of ω is set to be 0, and it is updated with the Adam optimizer using the learning rate of 10^{-4}. For image super resolution networks, we train the quantized models using the Adam optimizer for 3M batches with the batch size of 128. We use the learning rate of 10^{-5}. We set $\alpha = 5 \cdot 10^{-4}$. The initial value for ω is set to be 0 and it is updated by the Adam optimizer using the learning rate of 10^{-5}.

[4] `https://github.com/openimages/dataset`

[5] `https://github.com/BVLC/caffe`

Table 4: AlexNet quantization results on ImageNet classification.

Quantized layers	Weights	Activations	Top-1 / Top-5 accuracy (%)	
			Ours	DoReFa-Net (Zhou et al., 2016)*
Pretrained model	32-bit FLP	32-bit FLP	58.0 / 80.8	
(1) All layers	8-bit FXP	8-bit FXP	57.7 / 80.5	57.6 / 80.8
	4-bit FXP	4-bit FXP	56.5 / 79.4	56.9 / 80.3
	2-bit FXP	2-bit FXP	53.5 / 77.3	43.0 / 68.1
	1-bit FXP	8-bit FXP	52.2 / 75.8	47.5 / 72.1
		4-bit FXP	52.0 / 75.7	45.1 / 69.7
		2-bit FXP	50.5 / 74.6	43.6 / 68.3
		1-bit FXP	41.1 / 66.6	19.3 / 38.2
(2) Except the first and the last layers	8-bit FXP	8-bit FXP	57.7 / 80.6	57.5 / 80.7
	4-bit FXP	4-bit FXP	56.6 / 79.8	56.9 / 80.1
	2-bit FXP	2-bit FXP	54.1 / 77.9	53.1 / 77.3
	1-bit FXP	8-bit FXP	54.8 / 78.1	51.2 / 75.5
		4-bit FXP	54.8 / 78.2	51.9 / 75.9
		2-bit FXP	53.0 / 76.8	49.3 / 74.1
		1-bit FXP	43.9 / 69.0	40.2 / 65.5

* from our experiments using their code.

Table 5: SRCNN and CT-SRCNN (5-layer) quantization results for upscaling factor 3.

Model	Method	Weights	Activations	Set-14 PSNR (dB)	Set-14 SSIM	PSNR (dB) loss	SSIM loss
SRCNN 3-layer	Pretrained model	32-bit FLP	32-bit FLP	29.05	0.8161	-	-
	Ours	8-bit FXP	8-bit FXP	29.03	0.8141	0.02	0.0020
		4-bit FXP		28.99	0.8133	0.06	0.0028
		2-bit FXP		28.72	0.8075	0.33	0.0086
		1-bit FXP		28.53	0.8000	0.52	0.0161
	Gysel et al. (2018)*	8-bit FXP	8-bit FXP	28.58	0.7827	0.46	0.0328
CT-SRCNN 5-layer	Pretrained model	32-bit FLP	32-bit FLP	29.56	0.8273	-	-
	Ours	8-bit FXP	8-bit FXP	29.54	0.8267	0.02	0.0006
		4-bit FXP		29.48	0.8258	0.08	0.0015
		2-bit FXP		29.28	0.8201	0.28	0.0072
		1-bit FXP		29.09	0.8171	0.47	0.0102
	Gysel et al. (2018)*	8-bit FXP	8-bit FXP	29.04	0.8111	0.53	0.0148
Bicubic	-	-	-	27.54	0.7742	-	-

* from our experiments using their code at `https://github.com/pmgysel/caffe`.

B.2 Additional experimental results

In Table 4, we compare our quantization method to DoReFa-Net (Zhou et al., 2016) for the AlexNet model in Krizhevsky et al. (2012). The DoReFa-Net results in Table 4 are (re-)produced by us from their code[6]. We evaluate two cases where (1) all layers are quantized, and (2) all layers except the first and the last layers are quantized. The results in Table 4 show that 4-bit quantization is needed for accuracy loss less than 1%. For binary weights, we observe some accuracy loss of more or less than 10%. However, we can see that our quantization scheme performs better than DoReFa-Net in particular for low-precision cases, where the quantization error is larger and the mismatch problem of the forward and backward passes is more severe.

We also provide additional experimental results for image super resolution networks in Table 3. The experimental results show that our method successfully yields low-precision models of 8-bit weights and activations at negligible loss, which is better than Ristretto (Gysel et al., 2018).

[6] `https://github.com/ppwwyyxx/tensorpack/tree/master/examples/DoReFa-Net`