Progressive DNN Compression: A Key to Achieve Ultra-High Weight Pruning and Quantization Rates using ADMM

Anonymous ICCV submission

Paper ID 4501

Abstract

Weight pruning and weight quantization are two important categories of DNN model compression. Prior work on these techniques are mainly based on heuristics. A recent work developed a systematic framework of DNN weight pruning using the advanced optimization technique ADMM (Alternating Direction Methods of Multipliers), achieving one of state-of-art in weight pruning results. In this work, we first extend such one-shot ADMM-based framework to guarantee solution feasibility and provide fast convergence rate, and generalize to weight quantization as well. We have further developed a multi-step, progressive DNN weight pruning and quantization framework, with dual benefits of (i) achieving further weight pruning/quantization thanks to the special property of ADMM regularization, and (ii) reducing the search space within each step. Extensive experimental results demonstrate the superior performance compared with prior work. Some highlights: (i) we achieve 246×, 36×, and 8× weight pruning on LeNet-5, AlexNet, and ResNet-50 models, respectively, with (almost) zero accuracy loss; (ii) even a significant 61× weight pruning in AlexNet (ImageNet) results in only minor degradation in actual accuracy compared with prior work; (iii) we are among the first to derive notable weight pruning results for ResNet and MobileNet models; (iv) we derive the first lossless, fully binarized (for all layers) LeNet-5 for MNIST and VGG-16 for CIFAR-10; and (v) we derive the first fully binarized (for all layers) ResNet for ImageNet with reasonable accuracy loss. Our models and sample codes are released in anonymous link https://bit.ly/2TYx7Za.

1. Introduction

Deep neural networks (DNNs) are both computationally and storage intensive [15, 24]. A number of prior work have focused on developing *model compression* techniques for DNNs. These techniques, which are applied during the training phase of the DNN, aim to simultaneously reduce the model size (thus, the storage requirement) and accelerate the computation for inference – all these to be achieved with minor classification accuracy (or prediction quality) loss. Indeed the accuracy of a DNN inference engine after model compression is typically higher than that of a shallow neural network with no compression [9, 26]. Two important categories of DNN model compression techniques are *weight pruning* and *weight quantization*.

An early work on weight pruning of DNNs was done by Han *et al.* [9]. It is an iterative heuristic method, achieving a 9× reduction in the number of weights of AlexNet model (for ImageNet dataset). This weight pruning method has been extended in [4, 28, 8, 7, 26, 11] to either use more sophisticated algorithms to achieve a higher weight pruning rate, or to incorporate certain regularity or "structures" in the weight pruning framework. Weight quantization of DNNs has also been investigated in many recent work [16, 20, 33, 17, 27, 22, 14, 3]. Both storage and computational requirements of DNNs have been greatly reduced with tolerable accuracy loss. Indeed, multiplication operations (which are costly) may be eliminated when using binary, ternary, or power-of-2 weight quantizations [22, 14, 3].

To overcome the limitation of the highly heuristic nature in prior weight pruning work, a recent work [31] developed a systematic framework of DNN weight pruning using the advanced optimization technique ADMM (Alternating Direction Methods of Multipliers) [1, 13]. Through the adoption of ADMM, the original weight pruning problem is decomposed into two subproblems, one effectively solved using stochastic gradient descent as original DNN training, while the other solved optimally and analytically via Euclidean projection [31]. This method achieves one of state-of-art in weight pruning results, 21× weight reduction in AlexNet and 71.2× in LeNet-5 without accuracy loss. However, the direct application of ADMM technique lacks the guarantee on solution feasibility (satisfying all constraints) due to the non-convex nature of objective function (loss function), while there is also margin of improvement for solution quality (in terms of pruning rate under the same accuracy).

In this work, we first make the following extensions on the one-shot ADMM-based weight pruning [31]: (i) develop an integrated framework of dynamic ADMM regularization and masked mapping and retraining steps, thereby guaranteeing solution feasibility and providing high solution quality; (ii) incorporate the multi-ρ updating technique for faster (and better) ADMM convergence; and (iii) generalize to a unified framework applicable both the weight pruning and weight quantization. These extensions already provide higher performance than [31].

Beyond the above extensions, we observe the opportunity of performing further weight pruning from the results of the one-shot ADMM-based weight pruning framework. This is due to the special property of L_2-based ADMM regularization process. Similar observation also applies to the weight quantization problem, and both suggest a *progressive, multi-step model compression framework using ADMM*. In the progressive framework, the pruning/quantization results from the previous step serve as intermediate results and starting point for the subsequent step. It has an additional benefit of reducing the search space for weight pruning/quantization within each step. Detailed procedure and hyperparameter determination process have been carefully designed towards ultra-high weight pruning and quantization rates.

Extensive experimental results demonstrate that the proposed progressive framework consistently outperforms prior work. Some highlights: (i) we achieve 246×, 36×, and 8× weight pruning on LeNet-5, AlexNet, and ResNet-50 models, respectively, with (almost) zero accuracy loss; (ii) even a significant 61× weight pruning in AlexNet (ImageNet) results in only minor degradation in actual accuracy compared with prior work; (iii) we are among the first to derive notable weight pruning results for ResNet and MobileNet models; (iv) we derive the first lossless, fully binarized (for all layers) LeNet-5 for MNIST and VGG-16 model for CIFAR-10; and (v) we derive the first fully binarized (for all layers) ResNet model for ImageNet with reasonable accuracy loss. Our models and sample codes are released in anonymous link `https://bit.ly/2TYx7Za`.

2. Related Work

Weight pruning. An early work of weight pruning is [9]. It uses a heuristic, iterative method to prune weights of small magnitudes and retrain the DNN. It achieves 9× reduction in the number of weights on AlexNet for ImageNet dataset without accuracy degradation. However, this work achieves relatively low compression rate (2.7× for AlexNet) on CONV layers, which are the key computational part in state-of-the-art DNNs [25, 10]. Besides, indices are needed, at least one per weight, to index the relative location of the next weight. This method has been extended in two directions. The first is to improve reduction in the number of weights by using more sophisticated heuristics, e.g., incorporating both weight pruning and growing [8], using L_1 regularization [26], or genetic algorithm [5]. The second is enhancing the actual implementation efficiency by deriving an effective tradeoff between accuracy and compression rate, e.g., the *energy-aware pruning* [29], and incorporating regularity in weight pruning, e.g., the *channel pruning* [11] and *structured sparsity learning* [26] approaches.

Weight quantization. This method leverages the inherent redundancy in the number of bits for weight representation. Many of the prior art work [16, 20, 33, 17, 27, 22, 14, 3] are directed at quantization of weights to binary values, ternary values, or powers of 2 to facilitate hardware implementations, with acceptable accuracy loss. The state-of-the-art techniques [3, 16] adopt an iterative quantization and retraining framework, with some degree of randomness incorporated into the quantization step. This method results in less than 3% accuracy loss on AlexNet for binary weight quantization [16].

3. Overall Framework of Progressive DNN Model Compression

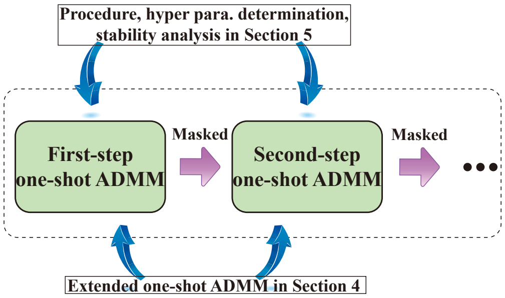

Procedure, hyper para. determination, stability analysis in Section 5
First-step one-shot ADMM
Masked
Second-step one-shot ADMM
Masked
•••
Extended one-shot ADMM in Section 4

Figure 1. Illustration of progressive DNN model compression.

Figure 1 illustrates the proposed progressive DNN weight pruning and weight quantization framework. The one-shot ADMM-based weight pruning or quantization is performed multiple times, each as a step in the progressive framework. The pruning/quantization results from the previous step serve as intermediate results and starting point for the subsequent step. As discussed before, the reasons to develop a progressive model compression framework are twofold: (i) The fact that many weights are close to zero after ADMM regularization enables further weight pruning (such observation also applies to quantization); and (ii) the multi-step procedure reduces the search space for weight pruning/quantization within each step.

Through extensive investigations, we conclude that a two-step progressive procedure will be in general sufficient for weight pruning and quantization, in which each step requires approximately the same number of training epochs as original DNN training. Further increase in the number of steps or the number of epochs in each step will result in only marginal improvement in the overall solution quality (e.g., 0.1%-0.2% accuracy improvement).

The detailed description of the proposed progressive framework will be presented in Section 4 and Section 5. Section 4 will present the proposed single-step, ADMM-based weight pruning and quantization framework, as an extension of [31] to guarantee solution feasibility and a generalization to weight quantization as well. Section 5 presents the motivation, detailed procedure, and hyperparameter determination of the proposed progressive model compression framework, along with illustration of why "progressive" is the key to ultra-high compression rates.

4. Single-Step, ADMM-based Weight Pruning and Quantization

4.1. Optimization Problem Formulation

Consider an N-layer DNN with both CONV and FC layers. The weights and biases of the i-th layer are respectively denoted by $\mathbf{W}_i$ and $\mathbf{b}_i$, and the loss function associated with the DNN is denoted by $f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big)$; see [31]. In this paper, $\{\mathbf{W}_i\}_{i=1}^N$ and $\{\mathbf{b}_i\}_{i=1}^N$ respectively characterize the collection of weights and biases from layer 1 to layer N. Then DNN weight pruning or weight quantization is formulated as the following optimization problem:

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big), \\ \text{subject to} \quad & \mathbf{W}_i \in \mathcal{S}_i, \ i = 1, \dots, N, \end{aligned} \tag{1}$$

For weight pruning, the constraint set is $\mathcal{S}_i = \big\{\mathbf{W}_i \big| \text{card}(\text{supp}(\mathbf{W}_i)) \le \alpha_i\big\}$, where 'card' refers to cardinality and 'supp' refers to the support set. Elements in $\mathcal{S}_i$ are $\mathbf{W}_i$ solutions, satisfying that the number of non-zero elements in $\mathbf{W}_i$ is limited by α_i for layer i. These α_i values are hyperparameters, with determination heuristic in Section 5. Besides the general, non-structured weight pruning scenario, the constraint set can be extended to incorporate specific "structures" corresponding to structured pruning techniques such as filter pruning, channel pruning, column pruning, etc., with detailed discussions in [32]. The appropriate structured pruning will facilitate high-parallelism implementations in hardware[1].

For weight quantization, elements in the constraint set $\mathcal{S}_i$ are $\mathbf{W}_i$ solutions, in which elements in $\mathbf{W}_i$ assume one of $q_{i,1}, q_{i,2}, ..., q_{i,M_i}$ values, where M_i denotes the number of these fixed values. Here, the $q_{i,j}$ values are *quantization levels* of weights of layer i in increasing order, and we focus on *equal-distance quantization* (the same distance between adjacent quantization levels) to facilitate hardware implementations. For the combination of weight pruning and quantization for DNNs, it is common practice to perform weight pruning first, and then quantization on the remaining, non-zero weights.

4.2. A Unified Solution Framework using ADMM

In problem (1) the constraint is combinatorial. As a result, this problem cannot be solved directly by stochastic gradient descent methods like original DNN training. However, the form of the combinatorial constraints on $\mathbf{W}_i$ is compatible with ADMM which is recently shown to be an effective method to deal with such clustering-like constraints [13, 18]

Despite such compatibility, there is still challenge in the direct application of ADMM due to the non-convexity in objective function. To overcome this challenge, we extend over [31] and develop a systematic framework of dynamic ADMM regularization and masked mapping and retraining steps. We can guarantee solution feasibility (satisfying all constraints) and provide high solution quality through this integration. This framework is unified and applies to both weight pruning and weight quantization, and will act as one step in the progressive DNN weight pruning/quantization framework.

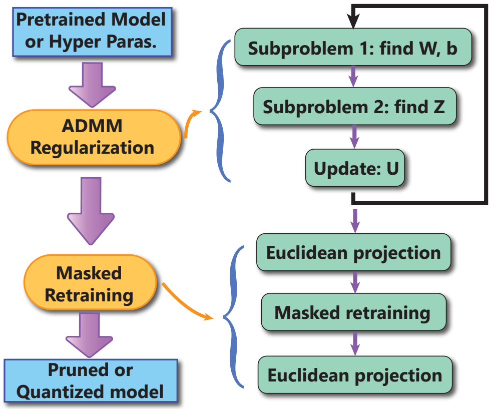


Figure 2. Procedure of one-shot ADMM-based weight pruning/quantization

ADMM Regularization Step: Corresponding to every set $\mathcal{S}_i$, $i = 1, \dots, N$ we define the indicator function $g_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathbf{W}_i \in \mathcal{S}_i, \\ +\infty & \text{otherwise.} \end{cases}$ Furthermore, we incorporate auxilliary variables $\mathbf{Z}_i$, $i = 1, \dots, N$. The original problem (1) is then equivalent to

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N g_i(\mathbf{Z}_i), \\ \text{subject to} \quad & \mathbf{W}_i = \mathbf{Z}_i, \ i = 1, \dots, N. \end{aligned} \tag{2}$$

Through formation of the augmented Lagrangian [1], the ADMM regularization decomposes problem (2) into

[1] The default weight pruning in this paper is the general, non-structured pruning. However, the proposed framework is also applicable to structured weight pruning, with results in supplementary materials.

two subproblems, and solves them iteratively until convergence[2]. The first subproblem is

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \ f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Z}_i^k + \mathbf{U}_i^k\|_F^2, \tag{3}$$

where $\mathbf{U}_i^k := \mathbf{U}_i^{k-1} + \mathbf{W}_i^k - \mathbf{Z}_i^k$. The first term in the objective function of (3) is the differentiable loss function of the DNN, and the second term is a quadratic regularization term of the $\mathbf{W}_i$'s, which is differentiable and convex. As a result (3) can be solved by stochastic gradient descent as original DNN training. Although we cannot guarantee the global optimality, it is due to the non-convexity of the DNN loss function rather than the quadratic term enrolled by our method. Please note that this first subproblem maintains the same form for weight pruning and quantization problems.

On the other hand, the second subproblem is given by

$$\underset{\{\mathbf{Z}_i\}}{\text{minimize}} \ \sum_{i=1}^N g_i(\mathbf{Z}_i) + \sum_{i=1}^N \frac{\rho_i}{2} \|\mathbf{W}_i^{k+1} - \mathbf{Z}_i + \mathbf{U}_i^k\|_F^2. \tag{4}$$

Note that $g_i(\cdot)$ is the indicator function of $\mathcal{S}_i$, thus this subproblem can be solved analytically and optimally [1]. For $i = 1, \ldots, N$, the optimal solution is the Euclidean projection of $\mathbf{W}_i^{k+1} + \mathbf{U}_i^k$ onto $\mathcal{S}_i$. For weight pruning, we can prove that the Euclidean projection results in keeping α_i elements in $\mathbf{W}_i^{k+1} + \mathbf{U}_i^k$ with the largest magnitudes and setting the remaining weights to zeros. For weight quantization, we can prove that the Euclidean projection results in mapping every element of $\mathbf{W}_i^{k+1} + \mathbf{U}_i^k$ to the quantization level closest to that element.

After both subproblems solved, we update the dual variables $\mathbf{U}_i$'s according to the ADMM rule [1] and thereby complete one iteration in ADMM regularization.

Increasing ρ in ADMM regularization: The ρ_i values are the most critical hyperparameter in ADMM regularization. We start from smaller ρ_i values, say $\rho_1 = \cdots = \rho_N = 1.5 \times 10^{-3}$, and gradually increase with ADMM iterations. This coincides with the theory of ADMM convergence [13, 18]. It in general takes 8 - 12 ADMM iterations for convergence (more iterations to converge for weight pruning and fewer for weight quantization), corresponding to 100 - 150 epochs in PyTorch. This convergence rate is comparable with the original DNN training.

Masked mapping and retraining: After ADMM regularization, we obtain intermediate $\mathbf{W}_i$ solutions. The subsequent step of masked mapping and retraining will guarantee the solution feasibility and improve solution quality. For weight pruning, the procedure is more straightforward. We first perform the said Euclidean projection (mapping) to guarantee that pruning constraints are satisfied. Next, we mask the zero weights and retrain the DNN with non-zero weights using training sets (while keeping the masked weights 0). In this way test accuracy (solution quality) can be (partially) restored, and solution feasibility (constraints) will be maintained.

For weight quantization, the procedure is more complicated. The reason is that the retraining process will affect the quantization results, thereby solution feasibility. To deal with this issue, we first perform Euclidean projection (mapping) of weights that are close enough (defined by a threshold value ϵ) to nearby quantization levels. Then we perform retraining on the remaining, unquantized weights (with quantized weights fixed) for accuracy improvement. Finally we perform Euclidean mapping on the remaining weights as well. In this way the solution feasibility will be guaranteed.

4.3. Explanation of Effectiveness in the Deep Learning Context

The proposed solution framework is different from the conventional utilization of ADMM, i.e., to accelerate the convergence of an originally convex problem [1, 12]. Rather, we integrate the ADMM framework with stochastic gradient descent. Aside from recent mathematical optimization results [13, 18] illustrating the advantage of ADMM with combinatorial constraints, the advantage of the proposed solution framework can be explained in the deep learning context as described below.

The proposed solution (3) can be understood as a smart, dynamic L_2 regularization method, in which the regularization target $\mathbf{Z}_i^k - \mathbf{U}_i^k$ will change judiciously and analytically in each iteration. On the other hand, conventional regularization methods (based on L_1, L_2 norms or their combinations) use a fixed regularization target, and the penalty is applied on all the weights. This will inevitably cause accuracy degradation. More illustrations of the ADMM-based dynamic regularization vs. conventional, fixed regularization will be provided in Section 5.3.

5. Progressive DNN Model Compression Framework: Detailed Procedure

5.1. Motivation

During the implementation of the one-shot weight pruning framework described in Section 4, we observe that there are a number of unpruned weights with values very close to zero. The reason is the L_2 regularization nature in ADMM regularization step, which tends to generate very small, non-zero weight values even when they are not pruned. As the remaining number of non-zero weights is already significantly reduced during weight pruning, simply mapping these small-value weights to zero will result in accuracy degradation. On the other hand, this motivates us to perform weight pruning (and quantization) in a multi-step, progres-

[2] The details of ADMM are presented in [1, 31]. We omit the details due to space limitation.

sive manner. For weight pruning, the weights that have been pruned in the previous step will be masked and only the remaining, non-zero weights will be considered in the subsequent step. For weight quantization, we perform quantization on the weights in a subset of layers, fix these quantization results, and quantize the remaining layers in the subsequent step.

A second motivation of the progressive framework is to reduce the search space for weight pruning/quantization within each step. After all, weight pruning and quantization problems are essentially combinatorial optimizations. Although recently demonstrated to generate superior results on this type of problems [13, 18], ADMM-based solution still has a superlinear increase of computational complexity as a function of solution space. As a result, the complexity becomes very high with ultra-high compression rates (i.e., very large search space) beyond what can be achieved in prior work. The progressive framework, on the other hand, can mitigate this limitation and reduce the total training time (to 2× or slightly higher than training time of the original DNN).

5.2. Detailed Procedure and Hyperparameter Determination

Through extensive investigations, we conclude that a two-step progressive procedure will be in general sufficient for weight pruning and quantization, in which each step requires approximately the same number of training epochs as original DNN training. We have conducted experiments on CIFAR-10 and ImageNet benchmarks (AlexNet and ResNet-18 models) on the relative accuracy of two-step procedure vs. three-step procedure, in which each step uses 120 epochs for training in PyTorch. The results show that three-step procedure only possesses marginal improvement in the overall solution quality, i.e., accuracy improvement no greater than 0.2%. This makes the additional training time not entirely worthwhile.

Hyperparameter Determination and Sensitivity Analysis: A very critical question is how to determine the hyperparameters, in a highly efficient and reliable manner. This problem is challenging for weight pruning, because we need to determine both the target overall pruning rate and specific pruning rate for each layer, both required in the ADMM-based solution. For quantization it becomes relatively straightforward, as the target number of quantization bits is typically pre-specified (binary, ternary, 2-bit, etc.) and the same number of quantization bits for all layers is in general preferred in hardware. The objective is to minimize accuracy loss. As a result, the two-step procedure of weight quantization can be performed as follows: the first step performs quantization on all the weights except for the first and last layers, while the second step performs quantization on these two layers. This is because quantization on these two layers has more significant impact on the overall accuracy.

Let us focus again on the hyperparameter determination heuristic for weight pruning problems. Experiments demonstrate that at least 2× to 3× improvement in overall pruning rate can be achieved compared with the prior work [9], under the same accuracy or without accuracy loss. Again at least 50% improvement in pruning rate can be achieved compared with the prior work of one-shot ADMM-based weight pruning [31]. As a result, a simple but effective hyperparameter determination method is as follows: We set the target overall pruning rate in the first ADMM-based weight pruning step to be around 1.5× compared with what can be achieved (without accuracy loss) in prior work [9], or to be slightly lower than the final result in [31]. The target overall pruning rate in the second step will be doubled compared with the first step, or even further increased if there is still margin of improvement. The per-layer pruning rate will be inherited from the results in prior work and increased proportionally. According to our experiments, the above heuristic will generate consistently higher pruning rates than prior work without accuracy loss.

We have further conducted two experiments to demonstrate the stability of hyperparameter (per-layer pruning rates) selection. Detailed experimental setup and results are provided in supplementary materials. The general conclusions are: (i) certain degree of variations in the per-layer pruning rates will have only minor impact on the overall accuracy under the ADMM-based framework; (ii) for very deep DNNs such as ResNet-50, uniform pruning rates for all layers will result in a reasonably good overall pruning results. These results demonstrate the robustness of the hyperparameter determination process.

Although the above discussions are based on the general, non-structured weight pruning, the above hyperparameter determination is also applicable to structured pruning.

5.3. Discussions and Illustration of Effectiveness through Weight Pruning

Using AlexNet model on ImageNet data set as an example, Figure 3 demonstrates the Top-5 accuracy loss vs. overall pruning rates using various methods, including our proposed progressive framework, our enhanced one-shot ADMM-based pruning, iterative pruning and retraining reported in [9], L_1 and L_2 fixed regularizations and projected gradient descent (PGD). Figure 4 demonstrates the absolute Top-5 accuracy. Please note that we use a baseline AlexNet model with 60.0% Top-1 accuracy and 82.2% Top-5 accuracy, both higher than prior work such as [9, 31] (57.2% Top-1 and 80.2% Top-5). This is to reflect the recent advances in DNN training in PyTorch. As a result, our definition of accuracy loss (or lossless) is compared with respect to the enhanced accuracy. In other words, we aim to surpass the prior methods in both absolute accuracy and relative accuracy loss values.

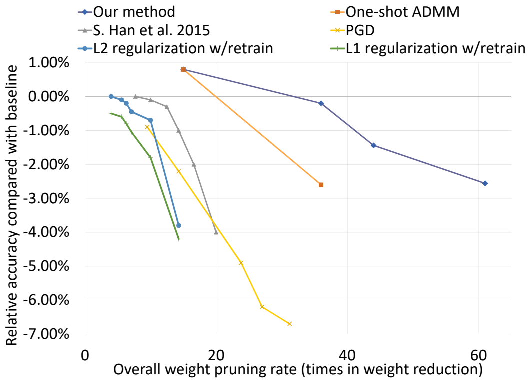


Figure 3. Relative top-5 accuracy compared with baseline for different pruning methods on AlexNet for ImageNet data set.

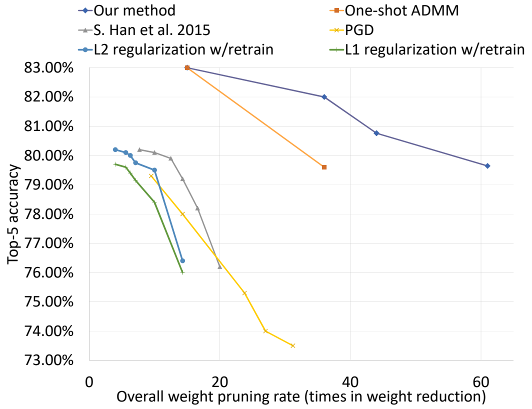
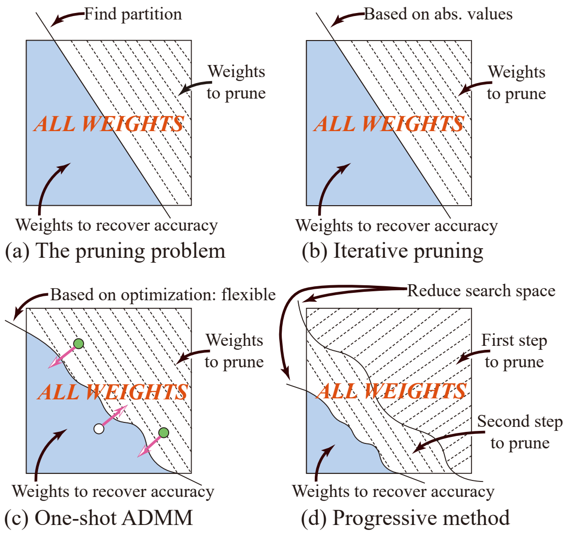


Figure 4. Absolute top-5 accuracy for different pruning methods on AlexNet for ImageNet data set.

We can clearly observe the performance ranking of these techniques. The proposed progressive framework outperforms all other methods. The second is one-shot ADMM-based pruning. The third is iterative pruning and retraining heuristic. And the last is fixed regularizations and PGD. We know from Section 4.3 that fixed regularizations and PGD suffer from penalizing all weights even if they are not pruned, thereby resulting in notable accuracy degradation. Then how to explain the performance gap among the other techniques?

To answer this question, we use Figure 5 as an illustration. The weight pruning problem can be understood as a *partitioning problem*, in which weights will be partitioned into two parts, one part all mapped to zero, while the other part utilized to restore accuracy. The straightforward iterative pruning method performs partitioning based only on the absolute values of the weights, smaller ones mapped to zero. The ADMM-based weight pruning method, on the other hand, allows partitioning using effective mathematical optimization methods, thereby achieving higher pruning rates without accuracy loss. Then new challenge exists on the high complexity in deriving such partitioning when the pruning rates become ultra-high, and this challenge can be effectively mitigated using the progressive method by reducing the search space within each step.



(a) The pruning problem
(b) Iterative pruning
(c) One-shot ADMM
(d) Progressive method

Figure 5. Illustration of effectiveness of the one-shot ADMM-based weight pruning and the progressive method.

6. Experimental Results

In this section, we evaluate the proposed progressive DNN model compression framework comprehensively, based on ImageNet ILSVRC-2012, CIFAR-10, and MNIST data sets, using AlexNet [15], VGGNet [24], ResNet-18/ResNet-50 [10], MobileNet V2 [23], and LeNet-5 DNN models, and comparing with various prior methods including single-shot ADMM. Our implementations are based on PyTorch, and the baseline accuracies are in many cases higher than those utilized in prior work, such as AlexNet and ResNet-50 for ImageNet, VGGNet and MobileNet V2 for CIFAR-10, etc. We conduct a fair comparison because we focus on the relative accuracy with our baseline instead of the absolute accuracy (which will of course outperform prior work).

Thanks to the compatibility of the proposed framework with DNN training, directly training a DNN model using the proposed framework has the same result as using a prior pre-trained DNN model. When a pre-trained DNN model is utilized, we limit the number of epochs in both steps in the progressive framework to be 120, similar to the original DNN training in PyTorch and much lower than the iterative pruning heuristic [9]. We use the hyperparameter determination procedure discussed in Section 5.3. The training and model compression are performed in PyTorch using NVIDIA 1080Ti, 2080, and Tesla P100 GPUs.

Due to space limitation, in this section we only present results on the general, non-structured weight pruning and sample results on binary quantizations. More comprehensive results on structured weight pruning, combination of

Table 1. Overall weight pruning rate comparisons on AlexNet model for ImageNet data set.

Method	Top-5 accuracy	Relative accuracy loss	Overall prun. rate
SVD [6]	79.4%	+0.9%	5.1×
Iter. prun. [9]	80.3%	−0.1%	9.1×
NeST [4]	80.3%	−0.1%	15.7×
Dyn. surg. [8]	80.0%	+0.2%	17.7×
One-shot ADMM [31]	80.2%	−0.0%	17.7×
Our one-shot	83.0%	−0.8%	15×
Our one-shot	79.6%	+2.6%	36×
Our method	82.0%	+0.2%	36×
Our method	80.8%	+1.4%	44×
Our method	79.7%	+2.5%	61×

Table 2. Convolutional layers weight pruning rate comparisons on the AlexNet model for ImageNet data set.

Method	Top-5 accuracy	Relative accuracy loss	Conv. prun. rate
Iter. prun. [9]	80.3%	−0.1%	2.7×
Dyn. surg. [8]	80.0%	+0.2%	3.1×
NeST [4]	80.3%	−0.1%	3.2×
Fine-grained [19]	80.4%	−0.2%	4.2×
L_1 method [26]	80.5%	−0.3%	5.0×
Our method	82.4%	−0.2%	8.6×
Our method	81.9%	+0.3%	11.2×

weight pruning and quantization, and convergence analysis are provided in the supplementary materials.

6.1. Experimental Results on Weight Pruning

6.1.1 Results on ImageNet Dataset

AlexNet Results: Table 1 compares the overall pruning rates of the whole AlexNet model (CONV and FC layers) vs. accuracy, between the proposed progressive framework and various prior methods. It can be clearly observed that the proposed framework outperforms prior methods, including the one-shot ADMM method [31]. With almost no Top-5 accuracy loss (note of our high baseline accuracy), we achieve 36× overall pruning rate. We achieve a notable 61× weight reduction with 79.7% Top-5 accuracy, just slightly below the baseline accuracy in prior work. We can clearly observe the advantage over one-shot ADMM method. With the same accuracy, the progressive framework achieves 61× weight reduction while our extended one-shot method achieves "only" 36×. This 36× in one-shot method has been derived using the same number of total training epochs as the progressive framework.

Table 2 compares the pruning rates on the CONV layers vs. Top-5 accuracy, since the CONV layers are the most computationally intensive in state-of-art DNNs. We achieve 8.6× pruning in CONV layers with even slight accuracy enhancement, and 11.2× pruning with minor accuracy loss, consistently outperforming prior work in CONV layer weight pruning.

VGG-16 Results: We conduct experiments on VGG-16 for ImageNet data set, with results similar to AlexNet. We achieve 34× overall weight reduction without accuracy loss, which is higher than 13× using iterative pruning [9], 15× in [30] or 19.9× using our extended one-shot ADMM (no corresponding results reported in [31]). Detailed table is omitted due to space limitation.

Table 3. Comparisons of overall weight pruning results on ResNet-50 for ImageNet data set.

Method	Top-5 Acc. Loss	Pruning rate
Uncompressed	0.0%	1×
Fine-grained [19]	0.1%	2.6×
Our one-shot	0.0%	4.5×
Our method	0.0%	8×
Our method	0.7%	17.4×

ResNet-18/ResNet-50 Results: We conduct experiments on ResNet-18 and ResNet-50 models for ImageNet data set. As there is lack of effective pruning results before, we conduct uniform weight pruning (the same pruning rate for all CONV and FC layers) to show the effectiveness with less optimized individual-layer pruning rates. The results are shown in Table 3. We achieve 8× overall pruning rate (also 8× pruning rate on CONV layers) on ResNet-50, without accuracy loss. We also achieve 6× overall pruning rate (also 6× pruning rate on CONV layers) on ResNet-18, without accuracy loss. These results clearly outperform the prior work which has limited overall pruning rate, which also did not mention CONV layer rate. It also outperforms our one-shot ADMM-based method, which achieves 4.5× uniform weight pruning on all layers (CONV and FC) on ResNet-50.

6.1.2 Results on CIFAR-10 Dataset

VGG-16 Results: We conduct experiments on VGG-16 results using the CIFAR-10 data set. The baseline accuracy is 93.7%, which is higher than those in prior work, e.g., 90.2% in [21] or 84.8% in [2]. We only present our results due to lack of prior work for fair comparison. We achieve 11.5× overall weight pruning without accuracy loss, or 40.3× with accuracy loss of 0.8%.

MobileNet V2 Results: We conduct experiments on MobileNet V2 results using the CIFAR-10 data set. The baseline accuracy is as high as 95.07% due to the adoption of mixup technique. We present our results in Table 4 due to lack of prior work for fair comparison. We achieve 5× weight pruning with almost no accuracy loss, starting from the high-accuracy baseline. We achieve 10× weight pruning (which is highly challenging for MobileNet) with only 1.3% accuracy loss.

Table 4. Our weight pruning results on MobileNet V2 for CIFAR-10 data set.

Method	Accuracy	Pruning rate
Uncompressed	95.07%	1×
Our method	95.49%	3.3×
Our method	94.90%	5×
Our method	94.70%	6.7×
Our method	93.75%	10×

6.1.3 Results on MNIST Dataset

Table 5 demonstrates the comparison results on LeNet-5 model using MNIST data set. Through the progressive framework, we achieve an unprecedented 246× overall weight reduction with almost no accuracy loss. It clearly outperforms one-shot ADMM (71.2× using prior one-shot ADMM [31] and 85× using our extended one-shot ADMM) and other prior methods. Please note that our extended one-shot ADMM-based method also slightly outperforms the prior counterpart [31].

Table 5. Comparisons of overall weight pruning results on LeNet-5 for MNIST data set.

Method	Accuracy	Pruning rate
Uncompressed	99.2%	1×
Network Pruning [9]	99.2%	12.5×
One-shot ADMM [31]	99.2%	71.2×
Optimal Brain Surg. [7]	98.3%	111×
Our one-shot	99.2%	85×
Our method	99.2%	200×
Our method	99.0%	246×

6.2. Sample Results on Weight Quantization

Binary Weight Quantization Results on LeNet-5: To the extent of authors' knowledge, we achieve the first lossless, fully binarized LeNet-5 model in which weights in all layers are binarized. The accuracy is still 99.21%, lossless compared with baseline. We do not list the comparison results due to limited space, but claim that our method already achieves the highest possible accuracy. We claim that becoming lossless is challenging even for MNIST. For example, recent work [2] results in 2.3% accuracy degradation on MNIST for full binarization, with baseline accuracy 98.66%.

Weight Quantization on CIFAR-10: We also achieve the first lossless, fully binarized VGG-16 for CIFAR-10, in which weights in all layers (including the first and the last) are binarized. The accuracy is 93.53%. We would like to point out that fully ternarized quantization results in 93.66% accuracy. Table 6 shows our results and comparisons.

Table 6. Comparisons of fully binary (ternary) weight quantization results on VGG-16 for CIFAR-10 data set.

Method	Accuracy	Num. of bits
Baseline of [2]	84.80%	32
8-bit [2]	84.07%	8
Binary [2]	81.56%	1
Our baseline	93.70%	32
Our ternary	93.66%	2 (ternary)
Our binary	93.53%	1

Binary Weight Quantization Results on ResNet for ImageNet Dataset: The binarization of ResNet models on ImageNet data set is widely acknowledged as a very challenging task. As a result, there are very limited prior work (e.g., the one-shot ADMM [16]) with binarization results on ResNet models. As [16] targets ResNet-18 (which is even more challenging than ResNet-50 or larger ones), we make a fair comparison on the same model. Table 7 demonstrates the comparison results (Top-5 accuracy loss). In prior work, it is by default that the first and last layers are not quantified (or quantized to 8 bits) as these layers have a significant effect on overall accuracy. When leaving the first and last layers unquantized, our framework is not progressive, but an extended one-shot ADMM-based framework. We can observe the higher accuracy compared with the prior method under this circumstance (first and last layers unquantized while the rest of layers binarized). The Top-1 accuracy has similar result: 3.8% degradation in our extended one-shot and 4.3% in [16].

Table 7. Comparisons of weight quantization results on ResNet-18 for ImageNet data set.

Method	Relative Top-5 acc. loss	Num. of bits
Uncompressed	0.0%	32
One-shot ADMM quantization [16]	2.9%	1 (32 for the first and last)
Our method (one-shot)	2.5%	1 (32 for the first and last)
Our method	5.8%	1

Using the progressive framework, we can derive a fully binarized ResNet-18, in which weights in all layers are binarized. The accuracy degradation is 5.8%, which is noticeable and shows that the full binarization of ResNet is a challenging task even under the progressive framework. We did not find prior work for comparison on this result.

7. Conclusion

In this work, we extended the prior one-shot ADMM-based framework and developed a multi-step, progressive DNN weight pruning and quantization framework, in which we achieve further weight pruning/quantization and provide faster convergence rate. We achieve 246×, 36×, and 8× weight pruning on LeNet-5, AlexNet, and ResNet-50 models, respectively, with (almost) zero accuracy loss. We also derive the first lossless, fully binarized (for all layers) LeNet-5 for MNIST and VGG-16 for CIFAR-10.

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. 1, 3, 4

[2] H.-P. Cheng, Y. Huang, X. Guo, Y. Huang, F. Yan, H. Li, and Y. Chen. Differentiable fine-grained quantization for deep neural network compression. *arXiv preprint arXiv:1810.10351*, 2018. 7, 8

[3] 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. 1, 2

[4] X. Dai, H. Yin, and N. K. Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *arXiv preprint arXiv:1711.02017*, 2017. 1, 7

[5] X. Dai, H. Yin, and N. K. Jha. Nest: a neural network synthesis tool based on a grow-and-prune paradigm. *arXiv preprint arXiv:1711.02017*, 2017. 2

[6] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 7

[7] X. Dong, S. Chen, and S. Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 4860–4874, 2017. 1, 8

[8] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016. 1, 2, 7

[9] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1, 2, 5, 6, 7, 8

[10] 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. 2, 6

[11] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017. 1, 2

[12] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017. 4

[13] M. Hong, Z.-Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016. 1, 3, 4, 5

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016. 1, 2

[15] 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. 1, 6

[16] 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. 1, 2, 8

[17] D. Lin, S. Talathi, and S. Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016. 1, 2

[18] S. Liu, J. Chen, P.-Y. Chen, and A. Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, pages 288–297, 2018. 3, 4, 5

[19] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922*, 2017. 7

[20] E. Park, J. Ahn, and S. Yoo. Weighted-entropy-based quantization for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[21] Z. Qin, F. Yu, C. Liu, and X. Chen. Demystifying neural network filter pruning. *arXiv preprint arXiv:1811.02639*, 2018. 7

[22] 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. 1, 2

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 6

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 6

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 2

[26] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016. 1, 2, 7

[27] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016. 1, 2

[28] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv preprint arXiv:1611.05128*, 2016. 1

[29] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6071–6079, 2017. 2

[30] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017. 7

[31] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang. A systematic dnn weight pruning framework

using alternating direction method of multipliers. *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3, 4, 5, 7, 8

[32] T. Zhang, K. Zhang, S. Ye, J. Li, J. Tang, W. Wen, X. Lin, M. Fardad, and Y. Wang. Adam-admm: A unified, systematic framework of structured weight pruning for dnns. *arXiv preprint arXiv:1807.11091*, 2018. 3

[33] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 1, 2