Automatic Neural Network Compression by Sparsity-Quantization Joint Learning: A Constrained Optimization-based Approach

Haichuan Yang[1], Shupeng Gui[1], Yuhao Zhu[1], and Ji Liu[2]

[1]Department of Computer Science, University of Rochester, Rochester, USA
[2]AI Platform, Ytech Seattle AI Lab, FeDA Lab, Kwai Inc., Seattle, USA

Abstract

Deep Neural Networks (DNNs) are applied in a wide range of usecases. There is an increased demand for deploying DNNs on devices that do not have abundant resources such as memory and computation units. Recently, network compression through a variety of techniques such as pruning and quantization have been proposed to reduce the resource requirement. A key parameter that all existing compression techniques are sensitive to is the compression ratio (e.g., pruning sparsity, quantization bitwidth) of each layer. Traditional solutions treat the compression ratios of each layer as hyper-parameters, and tune them using human heuristic. Recent researchers start using black-box hyper-parameter optimizations, but they will introduce new hyper-parameters and have efficiency issue. In this paper, we propose a framework to jointly prune and quantize the DNNs automatically according to a target model size without using any hyper-parameters to manually set the compression ratio for each layer. In the experiments, we show that our framework can compress the weights data of ResNet-50 to be 836× smaller without accuracy loss on CIFAR-10, and compress AlexNet to be 205× smaller without accuracy loss on ImageNet classification.

1. Introduction

Nowadays, Deep Neural Networks (DNNs) are being applied everywhere around us. Besides running inference tasks on cloud servers, DNNs are also increasingly deployed in resource-constrained environments today, ranging from embedded systems in micro aerial vehicle and autonomous cars to mobile devices such as smartphones and Augmented Reality headsets. In these environments, DNNs often operate under a specific resource constraint such as the model size, execution latency, and energy consumption. Therefore, it is critical to compress DNNs to run inference under given resource constraints while maximizing the accuracy.

Table 1: Comparison across different *automated* model compression methods.

Methods \ Features	Pruning	Quantization	Automated	End-to-end
AMC [14]	✓		✓	
HAQ [42]		✓	✓	
CLIP-Q [41]	✓	✓	✓	
Ours	✓	✓	✓	✓

In the past few years, various techniques have been proposed to compress the DNN models. Pruning and quantization are two of which most widely used in practice. Pruning demands the weights tensor to be sparse, and quantization enforces each DNN weight has a low-bits representation. These methods will compress the DNN weights in each layer and result in a compressed DNN having lower resource consumption. It has been shown that by appropriately setting the compression rate and performing fine-tuning, the compression could bring negligible accuracy drop [11].

Recent research works [49, 14, 42, 31] found that given the resource constraint, the accuracy of compressed DNNs can be further improved by tuning the compression ratio (i.e., sparsity or quantization bitwidth) for each layer. A fundamental question is: *how to find the optimal compression ratio, e.g., sparsity and/or bitwidth, for each layer in a way that meets a given resource budget*. Traditional DNN compression methods [11, 53, 15] set the compression ratio of each layer based on human heuristics. Since the compression ratios can be seen as hyper-parameters, the idea in recent research of using black-box optimization for hyper-parameter search can be directly adopted [41]. He et al. [14] applied reinforcement learning (RL) in DNN pruning by formulating the pruning ratio as a continuous action and the accuracy as the reward. Wang et al. [42] applied the similar formulation but used it for searching the quantization bitwidth of each layer. CLIP-Q [41] proposed a compression method which required the sparsity and quantization bitwidth to be set as

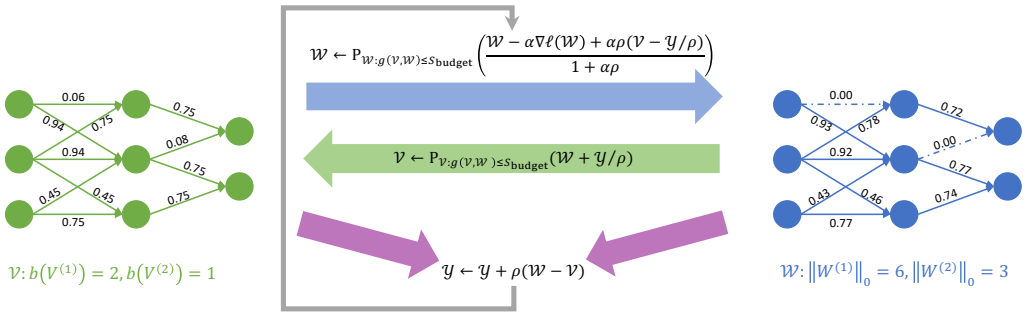


Figure 1: Illustration of the proposed DNN compression framework. DNN weight $\mathcal{W}$ is sparse and $\mathcal{V}$ is quantized. $\mathcal{V}$ is a "soft duplicate" of $\mathcal{W}$ and they are converged to be equal.

hyper-parameters, and they used Bayesian optimization libraries to search them. Evolutionary search (ES) was also being used in this scenario, for example, Liu et al. [31] used meta-learning and ES to find the pruning ratios of channel pruning. The basic idea of these methods was formulating the compression ratio search as a black-box optimization problem, but it introduced new hyper-parameters in the RL or ES algorithm. However, tuning black-box optimization algorithms could be very tricky [20] and usually inefficient [19]. Moreover, it introduces new hyper-parameters. For example, the RL algorithm DDPG [29] had dozens of hyper-parameters including batch size, actor / critic network architecture, actor /critic optimizer and learning rate, reward scale, discounting factor, reply buffer size, target network updating factor, exploration noise variance, and so on. Therefore, it is highly desirable to have an automated approach avoiding as much as possible the human heuristics.

Meanwhile, to maximize the compression performance, pruning and quantization could be applied simultaneously [11]. Thus, the layer-wise sparsity and quantization bitwidth will affect each other under this circumstance. For example, if layer i has larger bitwidth than layer j, then pruning layer i will contribute more than pruning layer j. Joint pruning and quantization increase the difficulty of manually choosing the compression ratios or hyper-parameter tuning.

In this paper, we present an end-to-end framework for automatic DNN compression. Our method can jointly prune and quantize the DNN model, and simultaneously learn the compression ratios and the compressed model weights. Instead of treating the compression ratios as hyper-parameters and using the black-box optimization, our method is based on a constrained optimization where an overall model size is set as the constraint to restrict the structure of the compressed model weights. Table 1 shows a comparison of our method with recently proposed automated model compression works.

The main contributions of this paper are summarized as follows:

- We propose an end-to-end framework to automatically compress DNNs without manually setting the compression ratio for each layer. It allows the user to set a budget and simultaneously utilizes pruning and quantization.
- We mathematically formulate the automated compression problem to a constrained optimization problem. The problem has a "sparse + quantized" constraint and it is further decoupled so that we can solve it using the Alternating Direction Method of Multipliers (ADMM) [1].
- The main challenge in using ADMM for the automated compression problem is solving the projection operators for pruning and quantization. We introduce the algorithms for getting the projection of the sparse constraint and quantization constraint. In the experiment, we validate our automated compression framework to show its superiority over the handcrafted and black-box hyper-parameter search methods.

2. Related Work

2.1. Model Compression Techniques

Due to the enormous impactions of mobile computing, more and more complicated DNN models are required to fit into those low-power consumption devices for real application. To solve the computation consumption issue onto the mobile systems, pruning and quantization are proposed as two practical approaches nowadays.

Pruning Pruning refers to decrease the amount of non-zero parameters in DNN models. Han et al. [12] proposed a simple approach by zeroing out the weights whose magnitudes are smaller than a threshold. By performing fine-tuning after removing the smaller weights, the accuracy drop is usually negligible even with a considerable compression ratio [11]. Besides using weights pruning for model compression, channel (filter / neuron) pruning [28, 57, 36, 16, 34, 59, 30, 51] was proposed to remove the entire filter of the CNN weights, thus also achieved inference acceleration. Wen et al. [44] introduced more sparsity structures into CNN pruning, such as shape-wise and depth-wise sparsity.

Quantization Besides decreasing the number of parameters with pruning, quantization is considered as another direction to compress DNNs. To relieve the cost of memory storage or computation, quantization focuses on converting the floating-point number elements to low-bits representations. For example, one can quantize all the parameters' precision from 32 bits to 8 bits or lower [11] to down-scale the model size. Extremely, the model weights can be binary [5, 37, 6, 18], or ternary [27, 58]. The quantization interval can be either uniform [21] or nonuniform [11, 35, 40, 55]. Typically, nonuniform quantization can achieve higher compression rate, while uniform quantization can provide acceleration. The quantization bitwidth could be further reduced by Hoffman coding [11, 4]. Besides the scalar quantization, vector quantization was also applied in DNN model compression [8, 45].

There are some methods performing training together with pruning and quantization, including Ye et al. [53] and CLIP-Q [41]. These methods relied on setting hyper-parameters to compress the layers with desired compression ratios, though the black-box hyper-parameter optimization method can be used [41]. Recently, ADMM was used to formulate and solve model compression problems [26, 56, 52, 38, 9]. However, These prior methods require the per-layer sparsity / bitwidth to be manually set. The main contribution of this paper is presenting an end-to-end framework to automatically prune and quantize DNNs without manually setting the compression ratio for each layer.

2.2. Automated Model Compression

Prior efforts on setting for the compression ratio of each layer mostly used either rule-based approaches [11, 17, 53, 15] or black-box hyper-parameter search. Rule-based approaches relied on heuristics, and thus were not optimal and unscalable as network architectures becoming more complex. Search-based approaches treated this problem as hyper-parameter search to eliminate the need for human labor. For pruning, NetAdapt [49] applied a greedy search strategy to find the sparsity ratio of each layer by gradually decreasing the resource budget and performing fine-tuning and evaluation iteratively. In each iteration, NetAdapt tried to reduce the number of nonzero channels of each layer, and picked the layer which results in smallest accuracy drop. Recent search-based approaches also employed reinforcement learning (RL), which used the accuracy and resource consumption to define the reward and guide the search to find pruning ratio [14] and quantization bitwidth [50, 42]. Guo et al. [10] used evolutionary search (ES) for network architecture search (NAS) and showed that it could be used for searching compression ratios. Liu et al. [31] used a hyper-network in the ES algorithm to find the layer-wise sparsity for channel pruning. Instead of regarding the layer-wise sparsity as hyper-parameters, recently proposed energy-constrained compression methods [47, 48] used optimization-based approaches to prune the DNNs under a given energy budget. Besides the above, there are some methods on searching efficient neural architectures [2, 39], while our work mainly concentrates on compressing a given architecture.

3. End-to-end Automated DNN Compression

In this section, we firstly introduce a general formulation of DNN compression, which is constrained by the total size of the compressed DNN weights. Secondly, we reformulate the original constraint to decouple the pruning and quantization and show the algorithm outline which uses ADMM to solve the constrained optimization. Lastly, as the proposed algorithm requires two crucial projection operators, we show that they can be formed as special integer linear programming (ILP) problems and introduce efficient algorithms to solve them.

3.1. Problem Formulation

Let $\mathcal{W} := \{W^{(i)}\}_{i=1}^{L}$ be the set of weight tensors of a DNN which has L layers. To learn a compressed DNN having a target size of S_{budget}, we have the constrained problem

$$\min_{\mathcal{W}} \ \ell(\mathcal{W}), \quad \text{s.t.} \ \sum_{i=1}^{L} b(W^{(i)}) \|W^{(i)}\|_0 \leq S_{\text{budget}}, \quad (1)$$

where $b(W)$ is the minimum bitwidth to encode all the nonzero elements of tensor W, i.e., $b(W) = \lceil \log_2 |\{\text{unique nonzero elements of } W\}| \rceil$. L_0-norm $\|W\|_0$ is the number of nonzero elements of W. The loss function ℓ is task-driven, for example, using the cross entropy loss as ℓ for classification, or mean squared error for regression.

Problem (1) is a general form of DNN compression. When assuming the bitwidth is fixed and same for all the layers, problem (1) reduces to the case of weights pruning [12]. When assuming the weight tensors are always dense, it is reduced to mixed-bitwidth quantization [42].

Compared with the ordinary training of deep learning, the compressed DNN learning problem (1) introduces a constraint, i.e. $\sum_{i=1}^{L} b(W^{(i)}) \|W^{(i)}\|_0 \leq S_{\text{budget}}$. It is defined by two non-differentiable functions $b(\cdot)$ and $\|\cdot\|_0$, which obstruct solving it via normal training algorithm. Although there is a projection-based algorithm which can handle the L_0-norm constraint, it can not be applied to our case because our constraint sums the products of $\|\cdot\|_0$ and $b(\cdot)$, which is more complicated.

3.2. Constraint Decoupling via Alternating Direction Method of Multipliers

We deal with the constraint in (1) by decoupling its L_0-norm and bitwidth parts. Specifically, we reformulate the

problem (1) to an equivalent form

$$\min_{\mathcal{W},\mathcal{V}} \ell(\mathcal{W}), \quad \text{s.t. } \mathcal{V} = \mathcal{W},\ g(\mathcal{V},\mathcal{W}) \le S_{\text{budget}}. \tag{2}$$

Where $\mathcal{V} := \{V^{(i)}\}_{i=1}^{L}$ is a duplicate of the DNN weights $\mathcal{W}$, and $g(\mathcal{V},\mathcal{W}) := \sum_{i=1}^{L} b(V^{(i)})\|W^{(i)}\|_0$.

In this paper, we apply the idea from ADMM to solve the above problem. We introduce the dual variable $\mathcal{Y} := \{Y^{(i)}\}_{i=1}^{L}$ and absorb the equality constraint into the augmented Lagrangian $\mathcal{L}_\rho(\mathcal{W},\mathcal{V},\mathcal{Y}) := \ell(\mathcal{W}) + \langle \mathcal{Y}, \mathcal{W}-\mathcal{V}\rangle + (\rho/2)\|\mathcal{W}-\mathcal{V}\|^2$, i.e.,

$$\min_{\mathcal{W},\mathcal{V}} \max_{\mathcal{Y}} \ell(\mathcal{W}) + \langle \mathcal{Y}, \mathcal{W}-\mathcal{V}\rangle + \frac{\rho}{2}\|\mathcal{W}-\mathcal{V}\|^2, \tag{3a}$$

$$\text{s.t. } g(\mathcal{V},\mathcal{W}) \le S_{\text{budget}}, \tag{3b}$$

where $\rho > 0$ is a hyper-parameter. Based on ADMM, we can solve this problem by updating $\mathcal{W}, \mathcal{V}$ and $\mathcal{Y}$ iteratively. In each iteration t, we have three steps corresponding to the variable $\mathcal{W}, \mathcal{V}$ and $\mathcal{Y}$ respectively.

Fix $\mathcal{V},\mathcal{Y}$, update $\mathcal{W}$. In this step, we treat $\mathcal{V},\mathcal{Y}$ as constants and update $\mathcal{W}$ to minimize $\mathcal{L}_\rho$, i.e., $\mathcal{W}^{t+1} =$

$$\begin{aligned}&\arg\min_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}} \ell(\mathcal{W}) + \langle \mathcal{Y}^t, \mathcal{W}-\mathcal{V}^t\rangle + \frac{\rho}{2}\|\mathcal{W}-\mathcal{V}^t\|^2 \\ =& \arg\min_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}} \ell(\mathcal{W}) + \frac{\rho}{2}\|\mathcal{W}-\mathcal{V}^t + \frac{1}{\rho}\mathcal{Y}^t\|^2. \end{aligned} \tag{4}$$

Because of the complexity of the DNN model and the large amount of the training data, $\ell(\cdot)$ is usually complex and the gradient based algorithms are often used to iteratively solve it. To support the gradient-based updating, we apply a proximal gradient method. Specifically, the loss function $\ell(\mathcal{W})$ is substituted with its first-order expansion, i.e., the problem (4) becomes

$$\begin{aligned}&\arg\min_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}} \ell(\mathcal{W}^t) + \langle \nabla\ell(\mathcal{W}^t), \mathcal{W}-\mathcal{W}^t\rangle \\ &\qquad + \frac{1}{2\alpha}\|\mathcal{W}-\mathcal{W}^t\|^2 + \frac{\rho}{2}\|\mathcal{W}-\mathcal{V}^t + \frac{1}{\rho}\mathcal{Y}^t\|^2 \\ =& \arg\min_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}} \left\|\mathcal{W}-\bar{\mathcal{W}}\right\|^2. \end{aligned} \tag{5}$$

Where $\bar{\mathcal{W}} := \frac{1}{1+\alpha\rho}(\mathcal{W}^t - \alpha\nabla\ell(\mathcal{W}^t) + \alpha\rho(\mathcal{V}^t - \frac{1}{\rho}\mathcal{Y}^t))$, $\nabla\ell(\mathcal{W}^t)$ is the (stochastic) gradient of ℓ at point $\mathcal{W}^t$, α is the learning rate, and . Problem (5) is the projection of $(\mathcal{W}^t - \alpha\nabla\ell(\mathcal{W}^t) + \alpha\rho(\mathcal{V}^t - \frac{1}{\rho}\mathcal{Y}^t))/(1+\alpha\rho)$ onto the set $\{\mathcal{W} : g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}\}$. We call it the compression projection with fixed bitwidth, and show how to solve it in Section 3.3.

Fix $\mathcal{W},\mathcal{Y}$, update $\mathcal{V}$. Here we use the updated $\mathcal{W}^{t+1}$ and minimize $\mathcal{L}_\rho$ in terms of $\mathcal{V}$.

$$\mathcal{V}^{t+1} = \arg\min_{\mathcal{V}: g(\mathcal{V},\mathcal{W}^{t+1}) \le S_{\text{budget}}} \|\mathcal{W}^{t+1} - \mathcal{V} + \frac{1}{\rho}\mathcal{Y}^t\|^2. \tag{6}$$

Since $\mathcal{W}^{t+1}$ and $\mathcal{Y}^t$ are fixed in this step, they can be seen as constants here. Problem (6) is the projection of $\mathcal{W}^{t+1} + \frac{1}{\rho}\mathcal{Y}^t$ onto $\{\mathcal{V} : g(\mathcal{V},\mathcal{W}^{t+1}) \le S_{\text{budget}}\}$. We call this projection the compression projection with fixed sparsity and leave the detail of solving it in Section 3.4.

Fix $\mathcal{W},\mathcal{V}$, update $\mathcal{Y}$. To update the dual variable $\mathcal{Y}$, we perform a gradient ascent step with learning rate as ρ:

$$\mathcal{Y}^{t+1} = \mathcal{Y}^t + \rho(\mathcal{W}^{t+1} - \mathcal{V}^{t+1}). \tag{7}$$

The above updating rules follow the standard ADMM. Recent theoretical analysis shows the convergence of ADMM also holds on non-convex problems [43]. In Section 4, we demonstrate these updating rules work well in our problem.

3.3. Compression Projection with Fixed Bitwidth

Problem (5) can be seen as a weighted L_0-norm projection $\mathrm{P}_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}}(\bar{\mathcal{W}})$ with $\bar{\mathcal{W}} = (\mathcal{W}^t - \alpha\nabla\ell(\mathcal{W}^t) + \alpha\rho(\mathcal{V}^t - \frac{1}{\rho}\mathcal{Y}^t))/(1+\alpha\rho)$:

$$\begin{aligned}\mathrm{P}_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{\text{budget}}}(\bar{\mathcal{W}}) :=& \arg\min_{\mathcal{W}} \|\mathcal{W}-\bar{\mathcal{W}}\|^2, \\ &\text{s.t. } \sum_{i=1}^{L} b(V^{t(i)})\|W^{(i)}\|_0 \le S_{\text{budget}}. \end{aligned} \tag{8}$$

We will show that this is actually a 0-1 Knapsack problem [46].

Proposition 1. *The projection problem in* (8) *is equivalent to the following 0-1 Knapsack problem:*

$$\max_{\mathcal{X} \text{ is binary}} \langle \bar{\mathcal{W}}^2, \mathcal{X}\rangle, \quad \text{s.t. } \langle \mathcal{A}, \mathcal{X}\rangle \le S_{budget}, \tag{9}$$

where $\mathcal{A}$ and $\mathcal{X}$ are of the same shape as $\bar{\mathcal{W}}$, and the elements of $A^{(i)}$ is defined as $A_j^{(i)} = b(V^{t(i)})$, $\forall j$. $\bar{\mathcal{W}}^2$ takes element-wise square of $\bar{\mathcal{W}}$. The optimal solution of (8) *is $P_{\mathcal{W}: g(\mathcal{V}^t,\mathcal{W}) \le S_{budget}}(\bar{\mathcal{W}}) = \mathcal{X}^* \odot \bar{\mathcal{W}}$, where $\mathcal{X}^*$ is the optimal solution to the knapsack problem* (9) *and $\odot$ is the element-wise multiplication.*

In this 0-1 Knapsack problem, $\bar{\mathcal{W}}^2$ is called the "profit", and $\mathcal{A}$ is the "weight". The 0-1 Knapsack is basically selecting a subset of items (corresponding to the DNN weights in our case) to maximize the sum of the profit and the total weight does not exceed the budget S_{budget}. The 0-1 Knapsack problem is NP hard, while there exists an efficient greedy algorithm [22] which works well in practice. The idea is based on the profit to weight ratio $(\bar{W}_j^{(i)})^2/A_j^{(i)}$. We sort all items based on this ratio and iteratively select the largest ones until the constraint boundary is reached. The theoretical complexity of this algorithm is $O(n\log(n))$, where n is the number of total items. Because the sorting and cumulative sum operations are supported on GPU, we can efficiently implement this algorithm on GPU and use it in our DNN compression framework.

3.4. Compression Projection with Fixed Sparsity

The solution of problem (6) is the projection $\mathrm{P}_{\mathcal{V}: g(\mathcal{V},\mathcal{W}^{t+1}) \le S_{\text{budget}}}(\mathcal{W}^{t+1} + \frac{1}{\rho}\mathcal{Y}^t)$, where the projection op-

erator $\mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W}^{t+1})\leq S_{\text{budget}}}(\cdot)$ is defined as

$$\mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W}^{t+1})\leq S_{\text{budget}}}(\bar{\mathcal{V}}) := \arg\min_{\mathcal{V}} \|\mathcal{V}-\bar{\mathcal{V}}\|^2, \quad (10)$$
$$\text{s.t. } \sum_{i=1}^{L} b(V^{(i)})\|W^{t+1(i)}\|_0 \leq S_{\text{budget}}.$$

The above problem can be also reformulate as an integer linear programming. In the following, we will introduce a special variant of Knapsack problem called Multiple-Choice Knapsack Problem (MCKP) [22] and show that the problem (10) can be written as an MCKP.

Definition 1. *Multiple-Choice Knapsack Problem (MCKP) [22]. Consider there are L mutually disjoint groups $G_1, ..., G_L$ which contain $n_1, ..., n_L$ items respectively. The j-th item from the i-th group has a "profit" ρ_{ij}, and "weight" ω_{ij}, $\forall i = 1, ..., L, j \in 1, ..., n_i$. MCKP formulates how to select exactly one item from each group to maximize the sum of profits and keep the sum of weights under a given budge β, i.e.,*

$$\max_{\mathbf{x} \text{ is binary}} \sum_{i=1}^{L}\sum_{j=1}^{n_i} \rho_{ij}\mathbf{x}_{ij},$$
$$s.t. \sum_{j=1}^{n_i} \mathbf{x}_{ij} = 1, \forall i = 1, ..., L; \sum_{i=1}^{L}\sum_{j=1}^{n_i} \omega_{ij}\mathbf{x}_{ij} \leq \beta.$$

Define $\mathcal{B}$ as the set of bitwidth candidates. In this paper, we use $\mathcal{B} = \{1, 2, 3, ..., 8\}$. Let $\mathcal{E}_j(\bar{V})$ be the error to quantize $\bar{V}$ with bitwidth j, i.e., $\mathcal{E}_j(\bar{V}) = \min_{V:b(V)=j} \|V - \bar{V}\|^2$, which can be solved by k-means algorithm for nonuniform quantization [11]. Now we are ready to reformulate the problem (10) as an MCKP.

Proposition 2. *The compression projection problem* (10) *can be reformulated to an instance of MCKP in Definition 1. Specifically, each group G_i is defined by each layer and has size $n_i = |\mathcal{B}|$. Each choice of the quantization bitwidth is regraded as an MCKP item. The profit ρ_{ij} is $-\mathcal{E}_j(\bar{V}^{(i)})$, the weight ω_{ij} is $j\|W^{t+1(i)}\|_0$, the Knapsack budget β is S_{budget}, and $\mathbf{x}_{ij}$ indicates selecting which bitwidth.*

The MCKP is also NP-hard. However, if we relax the binary constraints $\mathbf{x}_{ij} \in \{0, 1\}$ to $\mathbf{x}_{ij} \in [0, 1]$, it is reduced to a Linear Programming and can be solved efficiently. [54] transforms the linear relaxation of MCKP to the fractional knapsack problem and use a greedy algorithm to solve it. Based on this idea, we can get a feasible MCKP solution by the following steps:

1. For each group, sort the items based on their weights in ascending order, i.e., $\omega_{ij'} \geq \omega_{ij}$ if $j' \geq j$. According to [22, Proposition 11.2.2], the profits of the sorted items are nondecreasing, i.e., $\rho_{ij'} \geq \rho_{ij}$ if $\omega_{ij'} \geq \omega_{ij}$. The incremental profit density $(\rho_{ij} - \rho_{i,j-1})/(\omega_{ij} - \omega_{i,j-1})$ has descending order, i.e., $(\rho_{ij'} - \rho_{i,j'-1})/(\omega_{ij'} - \omega_{i,j'-1}) \leq (\rho_{ij} - \rho_{i,j-1})/(\omega_{ij} - \omega_{i,j-1})$ if $\omega_{ij'} \geq \omega_{ij}$.
2. Select the first item (having the smallest weight) of each group. It should be noted that the budget must be large enough to contain these items, otherwise there is no feasible solution under the constraints.
3. For other items, select the one with the largest incremental profit density. When selecting the j-th item of the i-th group, discard the $(j-1)$-th item. Repeat the same procedure for the 2nd, 3rd, ... largest ones, until the total weight of selected items exceeds the budget.

The above algorithm can find a feasible MCKP solution, i.e., selecting one item from each group and guarantee their total weight is under the given budget β. Its time complexity is $O(L|\mathcal{B}|\log(L|\mathcal{B}|))$. In practice, L and $|\mathcal{B}|$ are much smaller than the number of DNN weights, so the time complexity of this algorithm is negligible. The greedy solution has some nice properties and could be global optimal in some cases [22, Corollary 11.2.3]. By using the above algorithm to solve our compression projection problem (10), we can get the projection result of $\mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W}^{t+1})\leq S_{\text{budget}}}(\cdot)$, which essentially allocates the bitwidth across different layers.

We summarize the training procedure of our method in Algorithm 1. We use τ to denote the number of total SGD iterations of our algorithm. For large scale datasets, the number of SGD iterations could be very large. So we do not make the projections and dual update every time after we perform the proximal SGD on $\mathcal{W}$, but use a hyper-parameter τ' to control the frequency of dual updates. τ should be divisible by τ'. In our experiments, τ' is set to be the iteration number of one epoch, since we do not observe any improvement by using smaller τ'.

4. Experiments

In this section, we will evaluate our automated compression framework. We start with introducing the experiment setup such as evaluation and implementation details, then we show the compression results of our framework and compare it with state-of-the-art methods.

4.1. Experiment Setup

Datasets We evaluate our method on three datasets which are most commonly used in DNN compression: MNIST [25], CIFAR-10 [23], and ImageNet [7]. We use the standard training / testing data split and data preprocessing on all the three datasets. For ImageNet, we evaluate on the image classification task (1000 classes).

DNN models We evaluate on a wide range of DNN models, which are also used in current state-of-the-art compression methods. On MNIST, we use the LeNet-5 as in [11]. It has two convolution layers followed by two fully connected layers. For CIFAR-10, we evaluate on ResNet-20 and

Algorithm 1: Automatic DNN Compression.

Input: Original DNN parameterized $\mathcal{W}$, compression budget S_{budget}.

Result: The compressed DNN weights $\mathcal{W}^*$.

Initialize $\mathcal{W}$ with pretrained dense model, initialize $\mathcal{V}$ by uniformly quantizing $\mathcal{W}$, and initialize $\mathcal{Y} = \mathbf{0}$;

$\mathcal{W} \leftarrow \mathrm{P}_{\mathcal{W}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}})}(\mathcal{W})$;

$\mathcal{V} \leftarrow \mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W} + \frac{1}{\rho}\mathcal{Y})$;

$\mathcal{Y} \leftarrow \mathcal{Y} + \rho(\mathcal{W} - \mathcal{V})$;

for $t \leftarrow 1$ **to** τ **do**

  Compute stochastic gradient $\nabla\ell(\mathcal{W})$;

  $\mathcal{W} \leftarrow (\mathcal{W} - \alpha\nabla\ell(\mathcal{W}) + \alpha\rho(\mathcal{V} - \frac{1}{\rho}\mathcal{Y}))/(1 + \alpha\rho)$;

  if $t \pmod{\tau'} = 0$ **then**

    $\mathcal{W} \leftarrow \mathrm{P}_{\mathcal{W}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}})}(\mathcal{W})$;

    $\mathcal{V} \leftarrow \mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W} + \frac{1}{\rho}\mathcal{Y})$;

    $\mathcal{Y} \leftarrow \mathcal{Y} + \rho(\mathcal{W} - \mathcal{V})$;

  end

end

$\mathcal{W}^* = \mathcal{W}$.

ResNet-50 [13] which have 20 and 50 layers respectively. For ImageNet, we use the AlexNet [24] and the well-known compact model MobileNet [17]. In addition, we also investigate the compression performance of our method on the most recently proposed compact architectures MnasNet [39] and ProxylessNAS-mobile [2], which are searched by NAS algorithms.

Baselines and metric We compare our method with current state-of-the-art model compression methods related to ours. These methods include Recently proposed automated pruning methods AMC [14] and Constraint-Aware Compression [3]; Recently proposed automated quantization methods ReLeQ [50] and HAQ [42]; Methods which adopt both pruning and quantization: Deep Compression [11], Bayesian Compression [33], Ye et al. [53], and CLIP-Q [41].

Please refer to Tables 2, 3, and 4 for more detailed features of these methods. Although there are some overhead of the sparse index, we use the size of the compressed weights data to compute the compression rate since different indexing techniques may introduce unfairness in the comparison.

Implementation details We set the batch size as 256 for AlexNet and LeNet-5, and use 128 batch size on ResNets and MobileNet. We use the momentum SGD to optimize $\ell(\mathcal{W})$. We use initial learning rate α is set to 0.01 on AlexNet and MobileNet, and 0.1 on LeNet-5 and ResNets. We use the cosine annealing strategy [32] to decay the learning rate. We set the hyper-parameter $\rho = 0.05$ for all the experiments. To make a more clear comparison, the compression budget S_{budget} is set to be close to or smaller than the compared methods. Training is performed for 120 epochs on MNIST and CIFAR-10 and 90 epochs on ImageNet. Fine-tuning [11] is used on ImageNet for 60 epochs. To guarantee the final $\mathcal{W}^*$ satisfies the model size constraint, we directly perform a quantization to $\mathcal{W}^*$ with the bitwidth of $\mathcal{V}$.

4.2. Convergence and Sensitivity of ρ

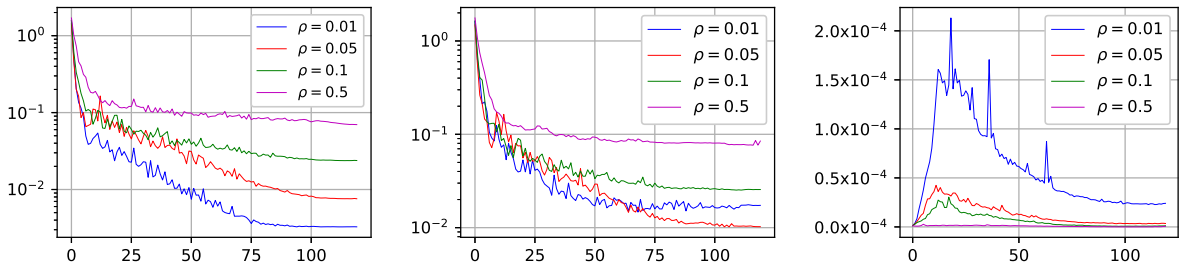
To address the impact of hyper-parameter ρ and the convergence of our training algorithm based on ADMM, we plot the training curves on MNIST classification experiments with various $\rho \in \{0.01, 0.05, 0.1, 0.5\}$ in Figure 2. Figure 2a shows the training loss of $\mathcal{W}$, Figure 2b shows the training loss of the quantized $\mathcal{W}$, where the bitwidth is set according to $\mathcal{V}$. We can see that $\ell(\mathcal{W})$ converges to smaller values with smaller ρ, since smaller ρ emphasizes more on the primal loss term. If perform the quantization on $\mathcal{W}$, the smallest loss is not achieved by the smallest ρ anymore, this is because $\mathcal{W}$ is not well constrained with the quantized structure when ρ is too small. To evaluate how the variables $\mathcal{W}$ differs from $\mathcal{V}$, we show the mean square error (MSE) between $\mathcal{W}$ and $\mathcal{V}$ in Figure 2c. We can see that the MSE curves usually increase in the beginning and then decrease, and $\rho = 0.05$ is enough to make the MSE $\rightarrow 0$.

4.3. Comparisons with State-of-the-arts

ImageNet In Table 2, we show the validation accuracies of compressed models of different methods on ImageNet classification. We list the nonzero weights percentage, averaged bitwidth, the compression rate (original weights size / compressed weights size), and the (top-1 / top-5) accuracy drop. For MobileNet, we compare with the quantization methods of Deep Compression [11] and HAQ [42]. We also compare with the uniform compression baselines [17]. The original MobileNet has 70.9% top-1 accuracy and 89.9% top-5 accuracy. Our quantization-only results with averaged bitwidth 2 and 3 have 7.1% and 1.19% top-1 accuracy drops respectively, which are about $2\times$ smaller than the HAQ counterparts (13.76% and 3.24%). The compression rate can be further improved to $26.7\times$ when jointly perform pruning and quantization.

For AlexNet, we compare with pruning or joint pruning and quantization methods. Unlike our end-to-end framework, all the compared methods set the pruning ratios and quantization bitwidth as hyper-parameters. Constraint-Aware Compression [3] and CLIP-Q [41] uses Bayesian optimization to choose these hyper-parameters, while others manually set them. The uncompressed AlexNet is from PyTorch pretrained models and has 56.52% top-1 accuracy and 79.07% top-5 accuracy. When compressing the model to be $118\times$ smaller, our method has an 1% top-1 accuracy improvement which is higher than the compressed CLIP-Q model with similar compression rate. Our method can also compress AlexNet to be $205\times$ smaller without accuracy drop, while the compressed model of Ye et al. [53] has a 0.1% top-1 accuracy drop with a similar compression rate.

For the NAS-based compact models, the uncompressed



(a) $\ell(\mathcal{W})$ (b) $\ell(\text{Quantized}(\mathcal{W}))$ (c) $\text{MSE}(\mathcal{W}, \mathcal{V})$.

Figure 2: Training loss and MSE (between $\mathcal{W}$ and $\mathcal{V}$) with different values of ρ.

Table 2: Comparison across different compression methods on ImageNet.

Model	Method	Automated	Pruning	Quantization	NZ%	Ave. bits	Comp. rate	Acc.-1↓	Acc.-5↓
MobileNet	Uniform Baseline [17]	✗	✓	✗	61%	-	1.6×	2.50%	1.70%
	Uniform Baseline [17]	✗	✓	✓	61%	8	6.6×	4.10%	2.90%
	Deep Compression [11]	✗	✗	✓	-	2	16×	33.28%	25.59%
	HAQ [42]	✓	✗	✓	-	2	16×	13.76%	8.03%
	Ours	✓	✗	✓	-	2	16×	7.10%	4.40%
	Deep Compression [11]	✗	✗	✓	-	3	10.7×	4.97%	3.05%
	HAQ [42]	✓	✗	✓	-	3	10.7×	3.24%	1.69%
	Ours	✓	✗	✓	-	3	10.7×	**1.19%**	**0.76%**
	Ours	✓	✓	✓	42%	2.8	**26.7**×	4.41%	2.61%
AlexNet	Constraint-Aware [3]	✓	✓	✗	4.9%	-	20×	2.57%	-
	Deep Compression [11]	✗	✓	✓	11%	5.4	54×	0.00%	-0.03%
	CLIP-Q [41]	✓	✓	✓	8%	3.3	119×	-0.70%	-
	Ours	✓	✓	✓	7.4%	3.7	118×	**-1.00%**	**-1.15%**
	Ye et al. [53]	✗	✓	✓	4%	4.1	**210**×	0.10%	-
	Ours	✓	✓	✓	5%	3.1	205×	-0.08%	-0.56%
MnasNet	Fixed-Bitwidth	✓	✓	✓	50%	4	16×	3.14%	1.86%
	Ours	✓	✓	✓	50%	3.7	17.1×	**1.66%**	**0.92%**
	Ours	✓	✓	✓	30%	3.0	**35.6**×	5.82%	3.23%
ProxylessNAS-mobile	Fixed-Bitwidth	✓	✓	✓	50%	4	16×	3.17%	1.73%
	Ours	✓	✓	✓	51%	3.8	16.8×	**2.13%**	**1.16%**
	Ours	✓	✓	✓	31%	2.9	**35.6**×	5.21%	2.84%

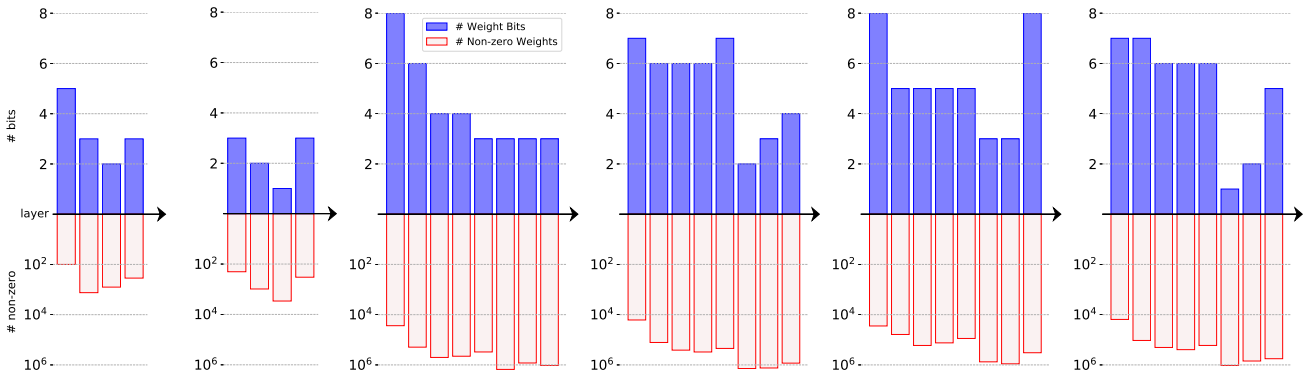
MnasNet has 73.46% top-1 accuracy and 91.51% top-5 accuracy, and the uncompressed ProxylessNAS-mobile has 74.59% top-1 accuracy and 92.20% top-5 accuracy. We also evaluated a joint pruning and quantization baseline (Fixed-Bitwidth) by fixing the bitwidth for all the layers as 4 and pruning 50% weights based on magnitude [11]. Compared with AlexNet, we can find that the accuracies on these compact models are easier to be influenced by compression. This phenomenon is similar as in MobileNet.

MNIST Table 3 shows the results of LeNet-5 on MNIST. The accuracy of the uncompressed LeNet-5 is 99.2%. Both Ye et al. [53] and our method can achieve about 2000× compression rate, while our compressed model does not have accuracy drop. Compare with the detail of its compressed model, we find that our method tends to leave more nonzero weights but uses less bits to represent each weight.

Table 3: Comparison across different compression methods on LeNet-5@MNIST. All the methods adopt both pruning and quantization.

Method	Automated	NZ%	Avg. bits	Comp. rate	Acc.↓
Deep Compression [11]	✗	8.3%	5.3	70×	0.1%
BC-GNJ [33]	✗	0.9%	5	573×	0.1%
BC-GHS [33]	✗	0.6%	5	771×	0.1%
Ye et al. [53]	✗	0.6%	2.8	1,910×	0.1%
Ours	✓	1.0%	1.46	**2,120**×	**0.0%**

CIFAR-10 Table 4 shows the results of the compressed ResNets on CIFAR-10 dataset. The accuracy of the original ResNet-20 is 91.29% and the accuracy of ResNet-50 is 93.55%. For ResNet-20, we compare with the automated quantization method ReLeQ [50]. For fair comparison, we evaluate two compressed models of our method, one only



(a) LeNet-5 ([53]) (b) LeNet-5 (Ours) (c) AlexNet (119×[41]) (d) AlexNet (Ours 118×) (e) AlexNet (210×[53]) (f) AlexNet (Ours 205×)

Figure 3: Visualization of the compressed results of different layers on LeNet-5 and AlexNet. The number of nonzero weights is shown in $\log_{10}$ scale. Our compressed models are presented in (b), (d) and (f) to compare with the network compressed by CLIP-Q [41] and Ye et al. [53].

uses quantization and another uses jointly pruning and quantization. For the quantization-only model, we achieve $16\times$ compression rate without accuracy drop, which has better accuracy and smaller size than ReLeQ. When introducing pruning, there is a 0.14% accuracy drop but the compression rate is improved to $35.4\times$.

Table 4: Comparison across different methods on CIFAR-10. All the methods automatically set the compression ratios.

Model	Method	Pruning	Quantization	NZ%	Ave. bits	Comp. rate	Acc.↓
ResNet-20	ReLeQ [50]	✗	✓	-	2.8	11.4×	0.12%
	Ours	✗	✓	-	2	16×	**0.00**%
	Ours	✓	✓	46%	1.9	**35.4**×	0.14%
ResNet-50	AMC [14]	✓	✗	60%	-	1.7×	-0.11%
	Ours	✓	✗	50%	-	2×	**-1.51**%
	Ours	✓	✓	4.2%	1.7	462×	-1.25%
	Ours	✓	✓	3.1%	1.9	565×	-0.90%
	Ours	✓	✓	2.2%	1.8	**836**×	0.00%

For ResNet-50, we compare with the automated pruning method AMC [14]. Its compressed ResNet-50 targeted on model size reduction has 60% of non-zero weights. In our experiment, we find that ResNet-50 still has a large space to compress. The pruning-only result of our method compress ResNet-50 with 50% weights and an 1.51% accuracy improvement. By performing jointly pruning and quantization, our method can compress the ResNet-50 with compression rate from $462\times$ to $836\times$. The accuracy loss is only met when compress the model to $836\times$ smaller, which suggests the ResNet-50 is mostly redundant on CIFAR-10 classification, and compressing it could reduce overfitting.

Compressed model visualization In Figure 3, we visualize the distribution of sparsity and bitwidth for each layer on LeNet-5 and AlexNet. Subfigures 3a, 3c and 3e show compressed models of Ye et al. [53] and CLIP-Q [41]. Subfigures 3b, 3d and 3f are our compressed models. For LeNet-5, we observe that our method preserves more nonzero weights in the third layer, while allocates less bitwidth compared with Ye et al. [53]. For AlexNet, our method has the trend of allocating larger bitwidth to convolutional layers than fully connected layers. CLIP-Q also allocates more bits to the convolutional layers, while Ye et al. [53] assign more bits to the first and last layer. Our method also shows a preference for allocating more bits to sparser layers. This coincides with the intuition that the weights of sparser layers may be more informative, and increasing the bitwidth on these layers also brings less storage growth.

5. Conclusion

As DNNs are increasing deployed on mobile devices, model compression is becoming more and more important in practice. Although many model compression techniques have been proposed in the past few years, lack of systematic approach to automatically set the layer-wise compression ratio diminishes their performance. Traditional methods require human labor to manually tune the compression ratios. Recent work uses black-box optimization to search the compression ratios but introduces instability of black-box optimization and is not efficient enough. We propose a constrained optimization formulation which considers both pruning and quantization and does not require compression ratio as hyper-parameter. By using ADMM, we build a framework to solve the constrained optimization problem efficiently. Experiment shows our method outperforms the handcrafted and hyper-parameter search approaches.

Acknowledgement

We gratefully acknowledge supports from NSF CCF Award #1714136.

References

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan 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.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[3] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 400–415, 2018.

[4] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016.

[5] 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*, pages 3123–3131, 2015.

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[8] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[9] Shupeng Gui, Haotao Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, pages 1283–1294, 2019.

[10] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] 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*, pages 1135–1143, 2015.

[13] 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*, pages 770–778, 2016.

[14] 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)*, pages 784–800, 2018.

[15] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[16] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[17] 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.

[18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[19] Alex Irpan. Deep reinforcement learning doesn't work yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

[20] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[21] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[22] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer, 2004.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[31] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *arXiv preprint arXiv:1903.10258*, 2019.

[32] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[33] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.

[34] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[35] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[36] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[37] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[38] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 925–938. ACM, 2019.

[39] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[40] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[41] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[42] 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*, pages 8612–8620, 2019.

[43] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

[44] 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*, pages 2074–2082, 2016.

[45] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Retraining and parameter sharing with harder cluster assignments for compressing deep convolutions. *arXiv preprint arXiv:1806.09228*, 2018.

[46] Haichuan Yang, Yuhao Zhu, and Ji Liu. Energy-constrained compression for deep neural networks via weighted sparse projection and layer input masking. *arXiv preprint arXiv:1806.04321*, 2018.

[47] Haichuan Yang, Yuhao Zhu, and Ji Liu. Ecc: Energy-constrained deep neural network compression via a bilinear regression model. *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] Haichuan Yang, Yuhao Zhu, and Ji Liu. Energy-constrained compression for deep neural networks via weighted sparse projection and layer input masking. In *International Conference on Learning Representations*, 2019.

[49] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[50] Amir Yazdanbakhsh, Ahmed T Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, and Hadi Esmaeilzadeh. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*, 2018.

[51] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[52] Shaokai Ye, Xiaoyu Feng, Tianyun Zhang, Xiaolong Ma, Sheng Lin, Zhengang Li, Kaidi Xu, Wujie Wen, Sijia Liu, Jian Tang, et al. Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm. *arXiv preprint arXiv:1903.09769*, 2019.

[53] Shaokai Ye, Tianyun Zhang, Kaiqi Zhang, Jiayu Li, Jiaming Xie, Yun Liang, Sijia Liu, Xue Lin, and Yanzhi Wang. A unified framework of dnn weight pruning and weight clustering/quantization using admm. *arXiv preprint arXiv:1811.01907*, 2018.

[54] Eitan Zemel. The linear multiple choice knapsack problem. *Oper. Res.*, 28(6):1412–1423, Dec. 1980.

[55] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, 2018.

[56] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[57] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[58] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[59] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.