LEARNING SPARSITY AND QUANTIZATION JOINTLY AND AUTOMATICALLY FOR NEURAL NETWORK COMPRESSION VIA CONSTRAINED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep Neural Networks (DNNs) are widely 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.

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 ratio (i.e., sparsity or quantization bitwidth) for each layer, the compression could bring negligible accuracy drop (Han et al., 2015a).

A fundamental question for these compression techniques 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 constraint*. Traditional DNN compression methods (Han et al., 2015a; Ye et al., 2018b; He et al., 2019) determine 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 (Tung & Mori, 2018). He et al. (2018) apply reinforcement learning (RL) in DNN pruning by formulating the pruning ratio as a continuous action and the accuracy as the reward. Wang et al. (2019) apply the similar formulation but uses it for searching the quantization bitwidth of each layer. CLIP-Q (Tung & Mori, 2018) propose a compression method which requires the sparsity and quantization bitwidth to be set as hyper-parameters, and they use

Table 1: Comparison across different automated model compression methods.

Methods \ Features	Support pruning	Support quantization	End-to-end optimization
AMC (He et al., 2018)	✓		
HAQ (Wang et al., 2019)		✓	
CLIP-Q (Tung & Mori, 2018)	✓	✓	
Ours	✓	✓	✓

Bayesian optimization libraries to search them. Evolutionary search (ES) is also being used in this scenario, for example, Guo et al. (2019) propose an ES-based network architecture search (NAS) method and use it for searching compression ratios. Liu et al. (2019) use meta-learning and ES to find the pruning ratios of channel pruning. The basic idea of these methods is formulating the compression ratio search as a black-box optimization problem, but it introduces new hyper-parameters in the RL or ES algorithm. However, tuning black-box optimization algorithms could be very tricky (Islam et al., 2017) and usually inefficient (Irpan, 2018). Moreover, it introduces new hyper-parameters. For example, the RL algorithm DDPG (Lillicrap et al., 2015) has 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 used together (Han et al., 2015a). Under this circumstance, a compression algorithm must tune both sparsity and quantization bitwidth for each layer. These two parameters will influence each other. For example, if layer i has higher bitwidth than another layer j, then pruning layer i (i.e., reducing the number of nonzero elements) will contribute more to model compression than pruning layer j. So jointly pruning and quantization increases the difficulty of manually choosing the compression ratios or hyper-parameter tuning.

In this paper, we present an end-to-end framework for automated DNN compression. Our method can jointly quantize and prune the DNN weights, and simultaneously learn the compression ratios (i.e. layer-wise sparsity and bitwidth) and 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.

Because our compression constraint considers both the bitwidth and sparsity of each layer, it is hard to be solved directly. In this work, we show that this constraint can be decoupled and the resultant formulation can be solved with the Alternating Direction Method of Multipliers (ADMM), which is widely used in constrained optimization (Boyd et al., 2011). Using ADMM to solve the proposed problem involves two projection operations, one is induced by pruning and another by quantization. We show that these two projection operators can be casted to integer linear programming (ILP) problems, and derive efficient algorithms to solve them. Figure 1 is an overview of the proposed framework, the main procedure consists of gradient descent / ascent and projection operations. For more details, refer to Section 3.

In summary, we make the following contributions:

- We propose an end-to-end framework to automatically compress DNNs without manually setting the compression ratio of each layer. It simultaneously utilizes pruning and quantization and directly learns the compressed DNN weights which can have different sparsity and bitwidth for each layer.
- 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 ADMM.
- 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

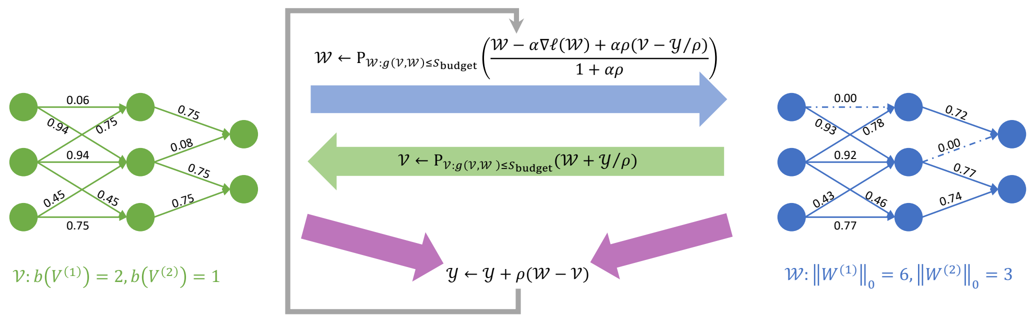

$\mathcal{W} \leftarrow \mathrm{P}_{\mathcal{W}:g(\mathcal{V},\mathcal{W})\le S_{\text{budget}}}\left(\frac{\mathcal{W} - \alpha\nabla\ell(\mathcal{W}) + \alpha\rho(\mathcal{V} - \mathcal{Y}/\rho)}{1+\alpha\rho}\right)$
$\mathcal{V} \leftarrow \mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W})\le S_{\text{budget}}}(\mathcal{W} + \mathcal{Y}/\rho)$
$\mathcal{Y} \leftarrow \mathcal{Y} + \rho(\mathcal{W} - \mathcal{V})$
0.06
0.94
0.75
0.94
0.45
0.45
0.75
0.75
0.08
0.75
0.75
$\mathcal{V}: b(V^{(1)}) = 2, b(V^{(2)}) = 1$
0.00
0.93
0.78
0.92
0.43
0.46
0.77
0.72
0.00
0.77
0.74
$\mathcal{W}: \|W^{(1)}\|_0 = 6, \|W^{(2)}\|_0 = 3$

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.

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. (2015b) 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 (Han et al., 2015a). Besides using weights pruning for model compression, channel (filter / neuron) pruning (Li et al., 2016b; Zhou et al., 2016; Molchanov et al., 2016; He et al., 2017; Luo et al., 2017; Zhuang et al., 2018; Liu et al., 2017; Ye et al., 2018a) is proposed to remove the entire filter of the CNN weights, thus also achieve inference acceleration. Wen et al. (2016) 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, we can quantize all the parameters' precision from 32 bits to 8 bits or lower (Han et al., 2015a) to down-scale the model size. Extremely, the model weights can be binary (Courbariaux et al., 2015; Rastegari et al., 2016; Courbariaux et al., 2016; Hubara et al., 2017), or ternary (Li et al., 2016a; Zhu et al., 2016). The quantization interval can be either uniform (Jacob et al., 2018) or nonuniform (Han et al., 2015a; Miyashita et al., 2016; Tang et al., 2017; Zhang et al., 2018). Typically, nonuniform quantization can achieve higher compression rate, while uniform quantization can provide acceleration. Besides the scalar quantization, vector quantization was also applied in DNN model compression (Gong et al., 2014; Wu et al., 2018).

There are some methods which perform training together with pruning and quantization, including Ye et al. (2018b) and CLIP-Q (Tung & Mori, 2018). These methods rely on setting hyper-parameters to compress the layers with desired compression ratios, although the black-box hyper-parameter optimization method can be used (Tung & Mori, 2018).

2.2 Automated Model Compression

Prior efforts on setting for the compression ratio of each layer mostly use either rule-based approaches (Han et al., 2015a; Howard et al., 2017; Ye et al., 2018b; He et al., 2019) or black-box hyper-parameter search. Rule-based approaches rely on heuristics, and thus are not optimal and unscalable as network architectures become more complex. Search-based approaches treat this problem as hyper-parameter search to eliminate the need for human labor. For pruning, NetAdapt (Yang et al., 2018) uses 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 tries to reduce the number of nonzero channels of each layer, and pick the layer which results in smallest accuracy drop. Recent search-based approaches also employ reinforcement learning (RL), which use the accuracy and resource consumption to define the reward and guide the search to find pruning ratio (He et al., 2018) and quantization bitwidth (Yazdanbakhsh et al., 2018; Wang et al., 2019). Guo et al. (2019) uses evolutionary search (ES) for network architecture search (NAS) and show that it can be used for searching compression ratios. Liu et al. (2019) use a hyper-network in the ES algorithm to find the layer-wise sparsity for channel pruning. Besides the above, there are some methods on searching efficient neural architectures (Cai et al., 2018; Tan et al., 2019), 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

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

Where $b(W)$ is the minimum bitwidth to encode all the nonzero elements of tensor W, i.e., $b(W) = \lceil \log_2 |\{\text{nonzero elements of } W\}| \rceil$. 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 (Han et al., 2015b). When assuming the weight tensors are always dense, it is reduced to mixed-bitwidth quantization (Wang et al., 2019).

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 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

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

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, i.e.,

$$\min_{\mathcal{W},\mathcal{V}} \max_{\mathcal{Y}} \mathcal{L}_\rho(\mathcal{W}, \mathcal{V}, \mathcal{Y}) := \ell(\mathcal{W}) + \sum_{i=1}^L \langle Y^{(i)}, W^{(i)} - V^{(i)} \rangle + (\rho/2) \sum_{i=1}^L \|W^{(i)} - V^{(i)}\|^2, \tag{3a}$$

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

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.,

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

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. Here we apply a proximal method to simplify the objective (4). Firstly, use a quadratic proxy to approximate $\ell(\mathcal{W})$, the problem (4) becomes

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

Where $\nabla \ell(\mathcal{W}^t)$ is the (stochastic) gradient of ℓ at point $\mathcal{W}^t$, and α is the learning rate. 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 $\mathcal{V}$.

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

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. Although ADMM relies on several assumptions (e.g., the objective function should be convex), we apply it in non-convex loss function and non-differentiable constraint functions to solve the minimax problem (3). In Section 4, we will 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 $\bar{\mathcal{W}} = (\mathcal{W}^t - \alpha \nabla \ell(\mathcal{W}^t) + \alpha\rho(\mathcal{V}^t - \frac{1}{\rho}\mathcal{Y}^t))/(1+\alpha\rho)$:

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

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

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 to approximately solve it (Kellerer et al., 2004). 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 greedy 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 operator $\mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W}^{t+1})\le S_{\text{budget}}}(\cdot)$ is defined as

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

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) (Kellerer et al., 2004) and show that the problem (10) can be written as an MCKP.

Definition 1. *Multiple-Choice Knapsack Problem (MCKP) (Kellerer et al., 2004). 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.,*

$$\max_{\mathbf{x}\text{ is binary}} \sum_{i=1}^{L}\sum_{j=1}^{n_i} \rho_{ij}\mathbf{x}_{ij}, \tag{11a}$$

$$\sum_{j=1}^{n_i} \mathbf{x}_{ij} = 1, \forall i = 1, ..., L; \ \sum_{i=1}^{L}\sum_{j=1}^{n_i} \omega_{ij}\mathbf{x}_{ij} \le \beta. \tag{11b}$$

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 (Han et al., 2015a). 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 $\|W^{t+1(i)}\|_0$, 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. Zemel (1980) transforms the linear relaxation of MCKP to the fractional knapsack problem and use a greedy algorithm to solve it. Here we can get a feasible MCKP solution based on the basic steps in Kellerer et al. (2004):

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 Kellerer et al. (2004, 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 $j-1$-the item. Repeat the same procedure for the 2nd, 3rd, ... largest ones, until the total weight of selected items exceeds the budget.

The above greedy 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. Although this greedy solution is not always global optimal, it has some nice properties and could be global optimal in some case (Kellerer et al., 2004, Corollary 11.2.3). By using MCKP-Greedy 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. Ideally, we should get $\mathcal{W}$ converged to $\mathcal{V}$ in the end, but t is hard to get $\mathcal{W}$ exactly equals to $\mathcal{V}$ in practice. So we perform a quantization to $\mathcal{W}$ to guarantee it satisfies the model size constraint.

Algorithm 1: End-to-end Automated DNN Compression Framework.

```
Input: Original DNN parameterized $\mathcal{W}$, compression budget $S_{\text{budget}}$, primal learning rate $\alpha$, dual
       learning rate $\rho$, iteration number $\tau$, and the dual updating interval $\tau'$.
Result: The compressed DNN weights $\mathcal{W}^*$.
1  Initialize $\mathcal{W}$ with pretrained dense model, initialize $\mathcal{V}$ by uniformly quantizing $\mathcal{W}$, and initialize
   $\mathcal{Y} = \mathbf{0}$;
2  $\mathcal{W} \leftarrow \mathrm{P}_{\mathcal{W}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W})$;
3  $\mathcal{V} \leftarrow \mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W} + \frac{1}{\rho}\mathcal{Y})$;
4  $\mathcal{Y} \leftarrow \mathcal{Y} + \rho(\mathcal{W} - \mathcal{V})$;
5  for $t \leftarrow 1$ to $\tau$ do
6      Sample data and compute stochastic gradient $\nabla\ell(\mathcal{W})$;
       $\mathcal{W} \leftarrow (\mathcal{W} - \alpha\nabla\ell(\mathcal{W}) + \alpha\rho(\mathcal{V} - \frac{1}{\rho}\mathcal{Y}))/(1 + \alpha\rho)$;
7      if $t \pmod{\tau'} = 0$ then
8          $\mathcal{W} \leftarrow \mathrm{P}_{\mathcal{W}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W})$;
9          $\mathcal{V} \leftarrow \mathrm{P}_{\mathcal{V}:g(\mathcal{V},\mathcal{W})\leq S_{\text{budget}}}(\mathcal{W} + \frac{1}{\rho}\mathcal{Y})$;
10         $\mathcal{Y} \leftarrow \mathcal{Y} + \rho(\mathcal{W} - \mathcal{V})$;
11     end
12 end
13 for $i \leftarrow 1$ to $L$ do
       /* In case $\mathcal{W}$ has not converged to $\mathcal{V}$ yet.          */
14     Quantize $W^{(i)}$ with the bitwidth $b(V^{(i)})$;
15 end
16 $\mathcal{W}^* = \mathcal{W}$.
```

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 (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009). 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[1]. It has two convolution layers followed by two fully connected layers. For CIFAR-10, we evaluate on ResNet-20 and ResNet-50 (He et al., 2016) which have 20 and 50 layers respectively. For ImageNet, we use the AlexNet (Krizhevsky et al., 2012) and the well-known compact model MobileNet (Howard et al., 2017).

Baselines and metric We compare our method with current state-of-the-art model compression methods related to ours. These methods include automated pruning method AMC (He et al., 2018), automated quantization methods ReLeQ (Yazdanbakhsh et al., 2018) and HAQ (Wang et al., 2019), and jointly pruning and quantization methods (Han et al., 2015a; Louizos et al., 2017; Tung & Mori, 2018; Ye et al., 2018b). Although there are some overhead of the sparse index, we only use the size of the compressed weights data to compute the compression rate because it directly corresponds to S_{budget}. In addition, 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(=0.9) 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 (Loshchilov & Hutter, 2016) 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 not larger than the compared methods. Training is performed for 120 epochs on LeNet-5 and ResNets and 90 epochs on AlexNet and MobileNet.

4.2 DNN Compression Results

MNIST In Table 2, we show the validation accuracies of compressed LeNet-5 models of different methods. We list the nonzero weights percentage, averaged bitwidth, and the compression rate (original weights size / compressed weights size), and the accuracy drop. The accuracy of the uncompressed LeNet-5 is 99.2%, and our method can achieve an impressive $2120\times$ compression rate without any accuracy drop. In the compared methods, the performance of Ye et al. (2018b) is the closest to our method. Compare with the detail of its compressed model, we find that our method tend to leave more nonzero weights but use less bits to represent each weight.

Table 2: Comparison across different compression methods on LeNet-5@MNIST.

Method	Nonzero%	Averaged bitwidth	Compression rate	Accuracy drop
Deep compression (Han et al., 2015a)	8.3%	5.3	70×	0.1%
BC-GNJ (Louizos et al., 2017)	0.9%	5	573×	0.1%
BC-GHS (Louizos et al., 2017)	0.6%	5	771×	0.1%
Ye et al. (2018b)	0.6%	2.8	1,910×	0.1%
Ours	1.0%	1.46	**2,120**×	**0.0%**

CIFAR-10 Table 3 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 (Yazdanbakhsh et al., 2018). For fair

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

comparison, we evaluate two compressed models of our method, one only uses quantization and another use 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 $35.4\times$.

For ResNet-50, we compare with the automated pruning method AMC (He et al., 2018). 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.

Table 3: Comparison across different compression methods on CIFAR-10.

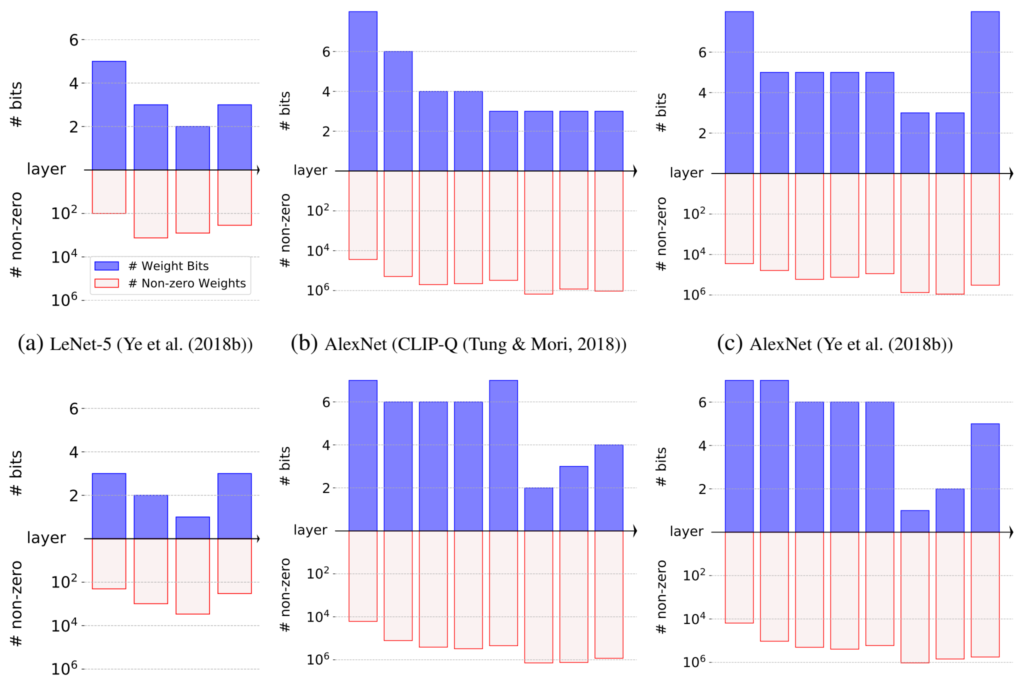
Model	Method	Nonzero%	Averaged bitwidth	Compression rate	Accuracy drop
ResNet-20	ReLeQ (Yazdanbakhsh et al., 2018)	-	2.8	11.4×	0.12%
	Ours	-	2	16×	**0.00**%
	Ours	46%	1.9	**35.4**×	0.14%
ResNet-50	AMC (He et al., 2018)	60%	-	1.7×	-0.11%
	Ours	50%	-	2×	**-1.51**%
	Ours	4.2%	1.7	462×	-1.25%
	Ours	3.1%	1.9	565×	-0.90%
	Ours	2.2%	1.8	**836**×	0.00%

ImageNet Table 4 shows the compressed results of MobileNet and AlexNet. For MobileNet, we compare with the quantization methods of Han et al. (2015a) and HAQ (Wang et al., 2019). The original MobileNet has 70.9% top-1 accuracy. Our quantization-only results with averaged bitwidth 2 and 3 have 7.1% and 1.19% 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 $16.7\times$ when jointly perform pruning and quantization.

For AlexNet, we compare with methods of joint pruning and quantization. Unlike our end-to-end framework, all the compared methods set the pruning ratios and quantization bitwidth as hyper-parameters. CLIP-Q (Tung & Mori, 2018) 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. When compressing the model to be $118\times$ smaller, our method has an 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. (2018b) has a 0.1% accuracy drop with a slightly lower compression rate.

Table 4: Comparison across different compression methods on ImageNet.

Model	Method	Nonzero%	Averaged bitwidth	Compression rate	Accuracy drop
MobileNet	Han et al. (2015a)	-	2	16×	33.28%
	HAQ (Wang et al., 2019)	-	2	16×	13.76%
	Ours	-	2	16×	7.10%
	Han et al. (2015a)	-	3	10.7×	4.97%
	HAQ (Wang et al., 2019)	-	3	10.7×	3.24%
	Ours	-	3	10.7×	**1.19%**
	Ours	42%	2.8	**26.7**×	4.41%
AlexNet	Han et al. (2015a)	11%	5.4	54×	0.00%
	CLIP-Q (Tung & Mori, 2018)	8%	3.3	119×	-0.70%
	Ours	7.4%	3.7	118×	**-1.00%**
	Ye et al. (2018b)	4%	4.1	193×	0.10%
	Ours	5%	3.1	**205**×	-0.08%



(a) LeNet-5 (Ye et al. (2018b)) (b) AlexNet (CLIP-Q (Tung & Mori, 2018)) (c) AlexNet (Ye et al. (2018b))

(d) LeNet-5 (Ours) (e) AlexNet (Our 118× compressed model) (f) AlexNet (Our 205× compressed model)

Figure 2: Visualization of the compressed results of different layers on LeNet-5 and AlexNet. The number of nonzero weights is shown in $\log_{10}$ scale.

Compressed model visualization In Figure 2, we visualize the distribution of sparsity and bitwidth for each layer on LeNet-5 and AlexNet. The first row shows compressed models of Ye et al. (2018b) and CLIP-Q (Tung & Mori, 2018), and the second row shows 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. (2018b). 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. (2018b) assigns 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 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. Different from prior work, we try to start from the root of the problem. Specifically, 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 can build the framework of our algorithm to solve the constrained optimization problem iteratively. Efficient algorithms for Knapsack problems are introduced to solve the sub-procedures in ADMM. In the future, we want to investigate other scenarios where we can substitute black-box optimization for more efficient optimization based approaches.

REFERENCES

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.

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

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

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.

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*, pp. 248–255. IEEE, 2009.

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

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.

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*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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*, pp. 4340–4349, 2019.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017.

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

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

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

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

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.

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*, pp. 2704–2713, 2018.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer, 2004. ISBN 978-3-540-40286-2.

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

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

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.

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

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

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.

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.

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*, pp. 2736–2744, 2017.

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

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

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*, pp. 5058–5066, 2017.

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

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.

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

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*, pp. 2820–2828, 2019.

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.

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*, pp. 7873–7882, 2018.

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

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

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

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. *arXiv preprint arXiv:1804.03230*, 2018.

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.

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*, 2018a.

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*, 2018b.

Eitan Zemel. The linear multiple choice knapsack problem. *Oper. Res.*, 28(6):1412–1423, December 1980. ISSN 0030-364X. doi: 10.1287/opre.28.6.1412. URL `https://doi.org/10.1287/opre.28.6.1412`.

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)*, pp. 365–382, 2018.

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

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

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*, pp. 875–886, 2018.