AutoCompress: An Automatic DNN Structured Pruning Framework for Ultra-High Compression Rates

Ning Liu,[1,2] Xiaolong Ma,[2] Zhiyuan Xu,[3] Yanzhi Wang,[2] Jian Tang,[1,3] Jieping Ye[1]
[1]DiDi AI Labs, [2]Northeastern University, [3]Syracuse University
{neilliuning, tangjian, yejieping}@didiglobal.com, ma.xiaol@husky.neu.edu, zxu105@syr.edu,
yanz.wang@northeastern.edu

Abstract

Structured weight pruning is a representative model compression technique of DNNs to reduce the storage and computation requirements and accelerate inference. An automatic hyperparameter determination process is necessary due to the large number of flexible hyperparameters. This work proposes AutoCompress, an automatic structured pruning framework with the following key performance improvements: (i) effectively incorporate the combination of structured pruning schemes in the automatic process; (ii) adopt the state-of-art ADMM-based structured weight pruning as the core algorithm, and propose an innovative additional purification step for further weight reduction without accuracy loss; and (iii) develop effective heuristic search method enhanced by experience-based guided search, replacing the prior deep reinforcement learning technique which has underlying incompatibility with the target pruning problem. Extensive experiments on CIFAR-10 and ImageNet datasets demonstrate that AutoCompress is the key to achieve ultra-high pruning rates on the number of weights and FLOPs that cannot be achieved before. As an example, AutoCompress outperforms the prior work on automatic model compression by up to 33× in pruning rate (120× reduction in the actual parameter count) under the same accuracy. Significant inference speedup has been observed from the AutoCompress framework on actual measurements on smartphone. We release models of this work at anonymous link: http://bit.ly/2VZ63dS.

1 Introduction

The high computational and storage requirements of large-scale DNNs, such as VGG (Simonyan and Zisserman 2015) or ResNet (He et al. 2016), make it prohibitive for broad, real-time applications at the mobile end. Model compression techniques have been proposed that aim at reducing both the storage and computational costs for DNN inference phase (Han et al. 2015; Wen et al. 2016; Guo, Yao, and Chen 2016; Min et al. 2018; Luo and Wu 2017; He, Zhang, and Sun 2017; He et al. 2018; Zhang et al. 2018a; 2018b; Min et al. 2018; Leng et al. 2018). One key model compression technique is DNN *weight pruning* (Wen et al. 2016; Luo and Wu 2017; Min et al. 2018; Guo, Yao, and Chen 2016; Han et al. 2015; He, Zhang, and Sun 2017; He et al. 2018; Zhang et al. 2018a; 2018b) that reduces the number of weight parameters, with minor (or no) accuracy loss.

There are mainly two categories of weight pruning. The general, *non-structured pruning* (Han et al. 2015; 2015; Luo and Wu 2017; Zhang et al. 2018a) can prune arbitrary weight in DNN. Despite the high pruning rate (weight reduction), it suffers from limited acceleration in actual hardware implementation due to the sparse weight matrix storage and associated indices (Han et al. 2015; Wen et al. 2016; He, Zhang, and Sun 2017). On the other hand, *structured pruning* (Wen et al. 2016; Min et al. 2018; He, Zhang, and Sun 2017; Zhang et al. 2018b) can directly reduce the size of weight matrix while maintaining the form of a full matrix, without the need of indices. It is thus more compatible with hardware acceleration and has become the recent research focus. There are multiple types/schemes of structured pruning, e.g., *filter pruning*, *channel pruning*, and *column pruning* for CONV layers of DNN as summarized in (Wen et al. 2016; Luo and Wu 2017; He, Zhang, and Sun 2017; Zhang et al. 2018b). Recently, a systematic solution framework (Zhang et al. 2018a; 2018b) has been developed based on the powerful optimization tool ADMM (Alternating Direction Methods of Multipliers) (Boyd et al. 2011; Suzuki 2013). It is applicable to different schemes of structured pruning (and non-structured one) and achieves state-of-art results (Zhang et al. 2018a; 2018b) by far.

The structured pruning problem of DNNs is flexible, comprising a large number of hyper-parameters, including the scheme of structured pruning and combination (for each layer), per-layer weight pruning rate, etc. Conventional hand-crafted policy has to explore the large design space for hyperparameter determination for weight or computation (FLOPs) reductions, with minimum accuracy loss. The trial-and-error process is highly time-consuming, and derived hyperparameters are usually sub-optimal. It is thus desirable to employ an automated process of hyperparameter determination for such structured pruning problem, motivated by the concept of AutoML (automated machine learning) (Zoph and Le 2016; Baker et al. 2016; Li et al. 2016; Liu et al. 2018a). Recent work AMC (He et al. 2018) employs the popular *deep reinforcement learning* (DRL) (Zoph and Le 2016; Baker et al. 2016) technique for automatic determination of per-layer pruning rates. However, it has limitations that (i) it employs an early weight pruning technique based on fixed regularization, and (ii) it only considers filter pruning for structured pruning. As we shall see later, the underlying incompatibility between the utilized DRL framework with the problem further limits its ability to achieve high weight pruning rates (the maximum reported pruning rate in (He et al. 2018) is only 5× and is non-structured pruning).

This work makes the following innovative contributions in the automatic hyperparameter determination process for DNN structured pruning. First, we analyze such automatic process in details and extract the *generic flow*, with four steps: (i) *action sampling*, (ii) *quick action evaluation*, (iii) *decision making*, and (iv) *actual pruning and result generation*. Next, we identify three sources of performance improvement compared with prior work. We adopt the ADMM-based structured weight pruning algorithm as the core algorithm, and propose an innovative additional purification step for further weight reduction without accuracy loss. Furthermore, we find that the DRL framework has underlying incompatibility with the characteristics of the target pruning problem, and conclude that such issues can be mitigated simultaneously using effective *heuristic search method* enhanced by experience-based guided search.

Combining all the improvements results in our automatic framework **AutoCompress**, which outperforms the prior work on automatic model compression by up to 33× in pruning rate (120× reduction in the actual parameter count) under the same accuracy. Through extensive experiments on CIFAR-10 and ImageNet datasets, we conclude that AutoCompress is the key to achieve ultra-high pruning rates on the number of weights and FLOPs that cannot be achieved before, while DRL cannot compete with human experts to achieve high pruning rates. Significant inference speedup has been observed from the AutoCompress framework on actual measurements on smartphone, based on our compiler-assisted mobile DNN acceleration framework. We release all models of this work at anonymous link: http://bit.ly/2VZ63dS.

2 Related Work

DNN Weight Pruning and Structured Pruning: DNN weight pruning includes two major categories: the general, *non-structured* pruning (Luo and Wu 2017; Guo, Yao, and Chen 2016; Han et al. 2015; Zhang et al. 2018a) where arbitrary weight can be pruned, and *structured* pruning (Wen et al. 2016; Luo and Wu 2017; Min et al. 2018; He, Zhang, and Sun 2017; Zhang et al. 2018b) that maintains certain regularity. Non-structured pruning can result in a higher pruning rate (weight reduction). However, as weight storage is in a sparse matrix format with indices, it often results in performance degradation in highly parallel implementations like GPUs. This limitation can be overcome in structured weight pruning.

Figure 1 illustrates three structured pruning schemes on the CONV layers of DNN: *filter pruning*, *channel pruning*, and *filter-shape pruning* (a.k.a. *column pruning*), removing whole filter(s), channel(s), and the same location in each filter in each layer. CONV operations in DNNs are commonly transformed to matrix multiplications by converting weight tensors and feature map tensors to matrices (Wen et al. 2016), named *general matrix multiplication* (GEMM). The key advantage of structured pruning is that a full matrix will be maintained in GEMM with dimensionality reduction, without the need of indices, thereby facilitating hardware implementations.

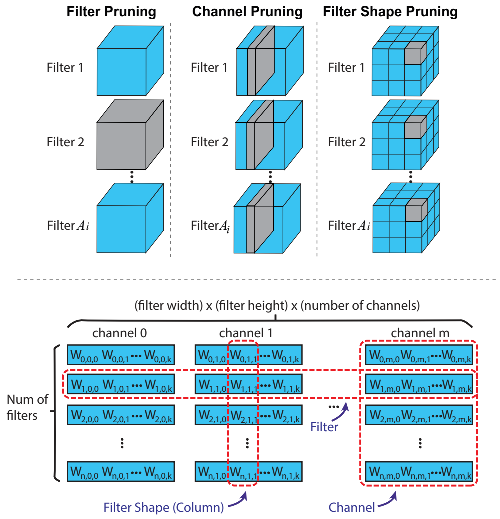
Filter Pruning
Channel Pruning
Filter Shape Pruning
Filter 1
Filter 2
Filter A_i
(filter width) x (filter height) x (number of channels)
channel 0
channel 1
channel m
Num of filters
$W_{0,0,0}$ $W_{0,0,1}$ ··· $W_{0,0,k}$
$W_{1,0,0}$ $W_{1,0,1}$ ··· $W_{1,0,k}$
$W_{2,0,0}$ $W_{2,0,1}$ ··· $W_{2,0,k}$
$W_{n,0,0}$ $W_{n,0,1}$ ··· $W_{n,0,k}$
$W_{0,1,0}$ $W_{0,1,1}$ ··· $W_{0,1,k}$
$W_{1,1,0}$ $W_{1,1,1}$ ··· $W_{1,1,k}$
$W_{2,1,0}$ $W_{2,1,1}$ ··· $W_{2,1,k}$
$W_{n,1,0}$ $W_{n,1,1}$ ··· $W_{n,1,k}$
$W_{0,m,0}$ $W_{0,m,1}$ ··· $W_{0,m,k}$
$W_{1,m,0}$ $W_{1,m,1}$ ··· $W_{1,m,k}$
$W_{2,m,0}$ $W_{2,m,1}$ ··· $W_{2,m,k}$
$W_{n,m,0}$ $W_{n,m,1}$ ··· $W_{n,m,k}$
Filter
Filter Shape (Column)
Channel

Figure 1: Different structured pruning schemes: A filter-based view and a GEMM view.

It is also worth mentioning that filter pruning and channel pruning are correlated (He, Zhang, and Sun 2017), as pruning a filter in layer i (after batch norm) results in the removal of corresponding channel in layer $i+1$. The relationship in ResNet (He et al. 2016) and MobileNet (Sandler et al. 2018) will be more complicated due to bypass links.

ADMM: Alternating Direction Method of Multipliers (ADMM) is a powerful mathematical optimization technique, by decomposing an original problem into two subproblems that can be solved separately and efficiently (Boyd et al. 2011). Consider the general optimization problem $\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x})$. In ADMM, it is decomposed into two subproblems on $\mathbf{x}$ and $\mathbf{z}$ ($\mathbf{z}$ is an auxiliary variable), to be solved iteratively until convergence. The first subproblem derives $\mathbf{x}$ given $\mathbf{z}$: $\min_{\mathbf{x}} f(\mathbf{x}) + q_1(\mathbf{x}|\mathbf{z})$. The second subproblem derives $\mathbf{z}$ given $\mathbf{x}$: $\min_{\mathbf{z}} g(\mathbf{z}) + q_2(\mathbf{z}|\mathbf{x})$. Both q_1 and q_2 are quadratic functions.

As a key property, ADMM can effectively deal with a subset of combinatorial constraints and yield optimal (or at least high quality) solutions. The associated constraints in DNN weight pruning (both non-structured and structured) belong to this subset (Hong, Luo, and Razaviyayn 2016). In DNN weight pruning problem, $f(\mathbf{x})$ is loss function of DNN and the first subproblem is DNN training with dynamic regularization, which can be solved using current gradient descent techniques and solution tools (Kingma and Ba 2014; Ten 2017) for DNN training. $g(\mathbf{x})$ corresponds to the combi-

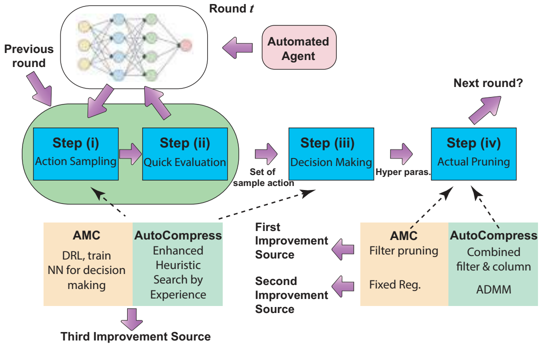

Figure 2: The generic flow of automatic hyperparameter determination framework, and sources of performance improvements.

natorial constraints on the number of weights. As the result of the compatibility with ADMM, the second subproblem has optimal, analytical solution for weight pruning via Euclidean projection. This solution framework applies both to non-structured and different variations of structured pruning schemes.

AutoML: Many recent work have investigated the concept of *automated machine learning* (AutoML), i.e., using machine learning for hyperparameter determination in DNNs. Neural architecture search (NAS) (Zoph and Le 2016; Baker et al. 2016; Liu et al. 2018a) is an representative application of AutoML. NAS has been deployed in Google's Cloud AutoML framework, which frees customers from the time-consuming DNN architecture design process. The most related prior work, AMC (He et al. 2018), applies AutoML for DNN weight pruning, leveraging a similar DRL framework as Google AutoML to generate weight pruning rate for each layer of the target DNN. In conventional machine learning methods, the overall performance (accuracy) depends greatly on the quality of features. To reduce the burdensome manual feature selection process, automated feature engineering learns to generate appropriate feature set in order to improve the performance of corresponding machine learning tools.

3 The Proposed AutoCompress Framework for DNN Structured Pruning

Given a pretrained DNN or predefined DNN structure, the automatic hyperparameter determination process will decide the per-layer weight pruning rate, and type (and possible combination) of structured pruning scheme per layer. The objective is the maximum reduction in the number of weights or FLOPs, with minimum accuracy loss.

3.1 Automatic Process: Generic Flow

Figure 2 illustrates the *generic flow* of such automatic process, which applies to both AutoCompress and the prior work AMC. Here we call a sample selection of hyperparamters an "action" for compatibility with DRL. The flow has the following steps: (i) *action sampling*, (ii) *quick action evaluation*, (iii) *decision making*, and (iv) *actual pruning and result generation*. Due to the high search space of hyperparameters, steps (i) and (ii) should be fast. This is especially important for step (ii), in that we cannot employ the time-consuming, retraining based weight pruning (e.g., fixed regularization (Wen et al. 2016; He, Zhang, and Sun 2017) or ADMM-based techniques) to evaluate the actual accuracy loss. Instead, we can only use simple heuristic, e.g., eliminating a pre-defined portion (based on the chosen hyperparameters) of weights with least magnitudes for each layer, and evaluating the accuracy. This is similar to (He et al. 2018). Step (iii) makes decision on the hyperparameter values based on the collection of action samples and evaluations. Step (iv) generates the pruning result, and the optimized (core) algorithm for structured weight pruning will be employed here. Here the algorithm can be more complicated with higher performance (e.g., the ADMM-based one), as it is only performed once in each round.

The overall automatic process is often iterative, and the above steps (i) through (iv) reflect only one round. The reason is that it is difficult to search for high pruning rates in one single round, and the overall weight pruning process will be progressive. This applies to both AMC and AutoCompress. The number of rounds is 4 - 8 in AutoCompress for fair comparison. Note that AutoCompress supports flexible number of progressive rounds to achieve the maximum weight/FLOPs reduction given accuracy requirement (or with zero accuracy loss).

3.2 Motivation: Sources of Performance Improvements

Based on the generic flow, we identify three sources of performance improvement (in terms of pruning rate, accuracy, etc.) compared with prior work. The **first** is the *structured pruning scheme*. Our observation is that an effective combination of filter pruning (which is correlated with channel pruning) and column pruning will perform better compared with filter pruning alone (as employed in AMC (He et al. 2018)). Comparison results are shown in the evaluation section. This is because of the high flexibility in column pruning, while maintaining the hardware-friendly full matrix format in GEMM. The **second** is the *core algorithm* for structured weight pruning in Step (iv). We adopt the state-of-art ADMM-based weight pruning algorithm in this step. Furthermore, we propose further improvement of a *purification step* on the ADMM-based algorithm taking advantages of the special characteristics after ADMM regularization. In the following two subsections, we will discuss the core algorithm and the proposed purification step, respectively.

The **third** source of improvement is the underlying principle of action sampling (Step (i)) and decision making (Step (iii)). The DRL-based framework in (He et al. 2018) adopts an exploration vs. exploitation-based search for action sampling. For Step (iii), it trains a neural network using action samples and fast evaluations, and uses the neural network to make decision on hyperparameter values. Our hypothesis is that DRL is inherently incompatible with the target automatic process, and can be easily outperformed by effective heuristic search methods (such as simulated annealing or genetic algorithm), especially the enhanced versions. More specifically, the DRL-based framework adopted in (He et al. 2018) is difficult to achieve high pruning rates (the maximum pruning rate in (He et al. 2018) is only 5× and is on non-structured pruning), due to the following reasons.

First, the sample actions in DRL are generated in a randomized manner, and are evaluated (Step (ii)) using very simple heuristic. As a result, these action samples and evaluation results (rewards) are just rough estimations. When training a neural network and relying on it for making decisions, it will hardly generate satisfactory decisions especially for high pruning rates. *Second*, there is a common limitation of reinforcement learning technique (both basic one and DRL) on optimization problem with constraints (Whiteson et al. 2011; Zhang et al. 2016; Henderson et al. 2018). As pruning rates cannot be set as hard constraints in DRL, it has to adopt a composite reward function with both accuracy loss and weight No./FLOPs reduction. This is the source of issue in controllability, as the relative strength of accuracy loss and weight reduction is very different for small pruning rates (the first couple of rounds) and high pruning rates (the latter rounds). Then there is the paradox of using a single reward function in DRL (hard to satisfy the requirement throughout pruning process) or multiple reward functions (how many? how to adjust the parameters?). *Third*, it is difficult for DRL to support flexible and adaptive number of rounds in the automatic process to achieve the maximum pruning rates. As different DNNs have vastly different degrees of compression, it is challenging to achieve the best weight/FLOPs reduction with a fixed, predefined number of rounds. These can be observed in the evaluation section on the difficulty of DRL to achieve high pruning rates. As these issues can be mitigated by effective heuristic search, we emphasize that an additional benefit of heuristic search is the ability to perform *guided search* based on prior human experience. In fact, the DRL research also tries to learn from heuristic search methods in this aspect for action sampling (Osband et al. 2016; Silver, Sutton, and Müller 2008), but the generality is still not widely evaluated.

3.3 Core Algorithm for Structured Pruning

This work adopts the ADMM-based weight pruning algorithm (Zhang et al. 2018a; 2018b) as the core algorithm, which generates state-of-art results in both non-structured and structured weight pruning. Details are in (Zhang et al. 2018a; 2018b; Boyd et al. 2011; Suzuki 2013). The major step in the algorithm is *ADMM regularization*. Consider a general DNN with loss function $f(\{\mathbf{W}_i\}, \{\mathbf{b}_i\})$, where $\mathbf{W}_i$ and $\mathbf{b}_i$ correspond to the collections of weights and biases in layer i, respectively. The overall (structured) weight pruning problem is defined as

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}}\ f(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}),\ \ \text{subject to}\ \mathbf{W}_i \in \mathcal{S}_i,\ \text{for all}\ i;$$

where $\mathcal{S}_i$ reflects the requirement that remaining weights in layer i satisfy predefined "structures". Please refer to (Wen et al. 2016; He, Zhang, and Sun 2017) for more details.

By defining (i) indicator functions $g_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathbf{W}_i \in \mathcal{S}_i \\ +\infty & \text{otherwise} \end{cases}$, (ii) incorporating auxiliary variable $\mathbf{Z}_i$ and dual variable $\mathbf{U}_i$, (iii) adopting augmented Lagrangian (Boyd et al. 2011), the ADMM regularization decomposes the overall problem into two subproblems, and iteratively solved them until convergence. The first subproblem is $\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}}\ f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N \frac{\rho_i}{2}\|\mathbf{W}_i - \mathbf{Z}_i^k + \mathbf{U}_i^k\|_F^2$. It can be solved using current gradient descent techniques and solution tools for DNN training. The second subproblem is $\underset{\{\mathbf{Z}_i\}}{\text{minimize}}\ \sum_{i=1}^N g_i(\mathbf{Z}_i) + \sum_{i=1}^N \frac{\rho_i}{2}\|\mathbf{W}_i^{k+1} - \mathbf{Z}_i + \mathbf{U}_i^k\|_F^2$, which can be optimally solved as Euclidean mapping.

Overall speaking, ADMM regularization is a dynamic regularization where the regularization target is dynamically adjusted in each iteration, without penalty on all the weights. This is the reason that ADMM regularization outperforms prior work of fixed L_1, L_2 regularization or projected gradient descent (PGD). To further enhance the convergence rate, the *multi-ρ method* (Ye et al. 2018) is adopted in ADMM regularization, where the ρ_i values will gradually increase with ADMM iterations.

3.4 Purification and Unused Weights Removal

After ADMM-based structured weight pruning, we propose the purification and unused weights removal step for further weight reduction without accuracy loss. First, as also noticed by prior work (He, Zhang, and Sun 2017), a specific filter in layer i is responsible for generating one channel in layer $i+1$. As a result, removing the filter in layer i (in fact removing the batch norm results) also results in the removal of the corresponding channel, thereby achieving further weight reduction. Besides this straightforward procedure, there is further margin of weight reduction based on the characteristics of ADMM regularization. As ADMM regularization is essentially a dynamic, L_2-norm based regularization procedure, there are a large number of non-zero, small weight values after regularization. Due to the non-convex property in ADMM regularization, **our observation** is that removing these weights can maintain the accuracy or even slightly improve the accuracy occasionally. As a result, we define two thresholds, a *column-wise threshold* and a *filter-wise threshold*, for each DNN layer. When the L_2 norm of a column (or filter) of weights is below the threshold, the column (or filter) will be removed. Also the corresponding channel in layer $i+1$ can be removed upon filter removal in layer i. Structures in each DNN layer will be maintained after this purification step.

These two threshold values are layer-specific, depending on the relative weight values of each layer, and the sensitivity on overall accuracy. They are hyperparameters to be determined for each layer in the AutoCompress framework, for maximum weight/FLOPs reduction without accuracy loss.

3.5 The Overall AutoCompress Framework for Structured Weight Pruning and Purification

In this section, we discuss the AutoCompress framework based on the enhanced, guided heuristic search method, in

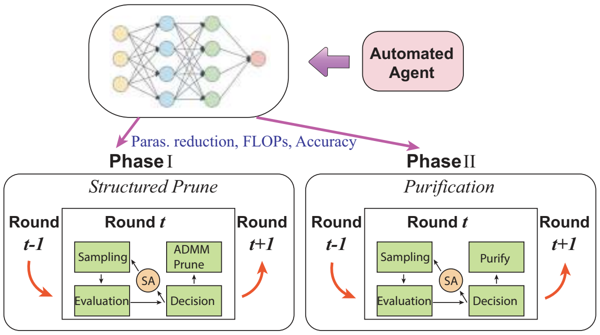

Figure 3: Illustration of the AutoCompress framework.

which the automatic process determines per-layer weight pruning rates, structured pruning schemes (and combinations), as well as hyperparameters in the purification step (discussed in Section 3.4). The overall framework has two phases as shown in Figure 3: *Phase I* for structured weight pruning based on ADMM, and *Phase II* for the purification step. Each phase has multiple progressive rounds as discussed in Section 3.1, in which the weight pruning result from the previous round serves as the starting point of the subsequent round. We use Phase I as illustrative example, and Phase II uses the similar steps.

The AutoCompress framework supports flexible number of progressive rounds, as well as hard constraints on the weight or FLOPs reduction. In this way, it aims to achieve the maximum weight or FLOPs reduction while maintaining accuracy (or satisfying accuracy requirement). For each round t, we set the overall reduction in weight number/FLOPs to be a factor of 2 (with a small variance), based on the result from the previous round. In this way, we can achieve around $4\times$ weight/FLOPs reduction within 2 rounds, already outperforming the reported structured pruning results in prior work (He et al. 2018).

We leverage a classical heuristic search technique *simulated annealing* (SA), with enhancement on *guided search* based on prior experience. The enhanced SA technique is based on the observation that a DNN layer with more number of weights often has a higher degree of model compression with less impact on overall accuracy. The basic idea of SA is in the search for actions: When a perturbation on the candidate action results in better evaluation result (Step (ii) in Figure 2), the perturbation will be accepted; otherwise the perturbation will be accepted with a probability depending on the degradation in evaluation result, as well as a temperature T. The reason is to avoid being trapped in local minimum in the search process. The temperature T will gradually decrease during the search process, in analogy to the physical "annealing" process.

Given the overall pruning rate $C_t \approx 2$ (on weight No. or FLOPs) in the current round, we initialize a randomized action A_t^0 using the following process: i) order all layers based on the number of remaining weights, ii) assign a randomized pruning rate (and partition between filter and column pruning schemes) for each layer, satisfying that a layer with more weights will have no less pruning rate, and iii) normalize the pruning rates by C_t. We also have a high initialized temperature T. We define *perturbation* as the change of weight pruning rates (and portion of structured pruning schemes) in a subset of DNN layers. The perturbation will also satisfy the requirement that the layer will more remaining weights will have a higher pruning rate. The result evaluation is the fast evaluation introduced in Section 3.1. The acceptance/denial of action perturbation, the degradation in temperature T, and the associated reduction in the degree of perturbation with T follow the SA rules until convergence. The action outcome will become the decision of hyperparameter values (Step (iii), this is different from DRL which trains a neural network). The ADMM-based structured pruning will be adopted to generate pruning result (Step (iv)), possibly for the next round until final result.

Algorithm 1 AutoCompress Framework for Structured Weight Pruning (similar process also applies to purification).

```
REQUIRE: Initial (unpruned) DNN model or DNN structure.
  for each progressive round $t$ do
      Initialize the action $A_t^0$ with partitioning of structured pruning schemes and pruning rate $\approx C_t$, satisfying the heuristic constraint.
      while $T >$ stop temperature do
          for iteration $i$ do
              Generate perturbation (magnitude decreases with $T$) on action, satisfying the heuristic constraint.
              Perform fast evaluation on the perturbation.
              if better evaluation result (higher accuracy) then
                  Accept the perturbation.
              else
                  Accept with probability $e^{-\frac{\Delta E}{T}}$, where $\Delta E$ is increase in evaluation cost (accuracy loss).
          Cool down $T \leftarrow \eta \cdot T$.
      The action outcome becomes the decision of hyperparameter values.
      Perform ADMM-based structured pruning to generate pruning result, for the next round.
```

4 Evaluation, Experimental Results, and Discussions

Setup: The effectiveness of AutoCompress is evaluated on VGG-16 and ResNet-18 on CIFAR-10 dataset, and VGG-16 and ResNet-18/50 on ImageNet dataset. We focus on the structured pruning on CONV layers, which are the most computationally intensive layers in DNNs and the major storage in state-of-art DNNs such as ResNet. In this section we focus on the objective function of reduction in the number of weight parameters. The implementations are based on PyTorch (Paszke et al. 2017). For structured pruning, we support (i) filter pruning only, and (ii) combined filter and column pruning, both supported in ADMM-based algorithm and AutoCompress framework. In the ADMM-based structured pruning algorithm, the number of epochs in each progressive round is 200, which is lower than the prior iterative pruning heuristic (Han et al. 2015). We use an initial penalty parameter $\rho = 10^{-4}$ for ADMM and initial learn-

Table 1: Comparison on pruning approaches using VGG-16 on CIFAR-10 Dataset

	Method	Accuracy	CONV Params Rt.	CONV FLOPs Rt.	Inference time
Original VGG-16		93.7%	1.0×	1.0×	14ms
Filter Pruning	2PFPCE (Min et al. 2018)	92.8%	4×	N/A	N/A
	2PFPCE (Min et al. 2018)	91.0%	8.3×	N/A	N/A
	ADMM, manual hyper. determ.	93.48%	9.3×	2.1×	7.1ms
Auto Filter Pruning	ADMM-based, enhanced SA	93.22%	13.7×	3.1×	4.8ms
	Train-From-Scratch	93.19%	13.7×	3.1×	4.8ms
	ADMM-based, enhanced SA	88.78%	47.4×	14.0×	1.7ms
Combined Structured Pruning	ADMM, manual hyper. determ.	93.26%	44.3×	8.1×	2.9ms
	Full **AutoCompress**	**93.21%**	52.2×	8.8×	2.7ms
	Train-From-Scratch	91.4%	52.2×	8.8×	2.7ms

Table 2: Comparison on pruning approaches using ResNet-18 (ResNet-50 in NISP and AMC) on CIFAR-10 Dataset

	Method	Accuracy	CONV Params Rt.	CONV FLOPs Rt.	Inference time
Original ResNet-18		93.9%	1.0×	1.0×	11ms
Filter Pruning	NISP (Yu et al. 2018)	93.2%	1.7×	N/A	N/A
	ADMM, manual hyper. determ.	93.9%	5.2×	2.7×	4.2ms
Auto Filter Pruning	AMC (He et al. 2018)	93.5%	1.7×	N/A	N/A
	ADMM-based, enhanced SA	93.91%	8.0×	4.7×	2.4ms
	Train-From-Scratch	93.89%	8.0×	4.7×	2.4ms
Combined Structured Pruning	ADMM, DRL hyper. determ.	93.55%	11.8×	3.8×	4.7ms
	ADMM, manual hyper. determ.	93.69%	43.3×	9.6×	1.9ms
	Full **AutoCompress**	93.43%	61.2×	13.3×	1.3ms
	Full **AutoCompress**	**93.81%**	54.2×	12.2×	1.45ms
	Train-From-Scratch	91.88%	54.2×	12.2×	1.45ms

ing rate 10^{-3}. The ADAM (Kingma and Ba 2014) optimizer is utilized. In the SA setup, we use cooling factor $\eta = 0.7$ and Boltzmann's constant $k = 10^{-3}$. The initial probability of accepting high energy (bad) moves is set to be relatively high.

Models and Baselines: We aim at fair and comprehensive evaluation on the effectiveness of three sources of performance improvements discussed in Section 3.2. Besides the original, unpruned DNN models, we compare with a set of prior baseline methods. Perhaps for software implementation convenience, almost all baseline methods we can find focus on filter/channel pruning. For fair comparison, we also provide pruning results on ADMM-based filter pruning with manual hyperparameter determination. This case is only different from prior work by a single source of performance improvement – the core algorithm using ADMM. We also show the results on ADMM-based filter pruning with enhanced SA-based hyperparameter determination, in order to show the effect of an additional source of improvement.

Beyond filter pruning only, we show the combined structured pruning results using ADMM to demonstrate the last source of performance improvement. We provide results on manual, our crafted DRL-based, and enhanced SA-based hyperparameter determination for fair comparison, the last representing the full version of AutoCompress. We provide the inference time of the pruned models using the latest Qualcomm Adreno 640 GPU in Samsung Galaxy S10 smartphone. The results clearly demonstrate the actual acceleration using the combined structured pruning. Note that our mobile DNN acceleration framework is a compiler assisted, strong framework by itself. For the original VGG-16 and ResNet-18 (without pruning) on CIFAR-10, it achieves 14ms and 11ms end-to-end inference times, respectively, on the Adreno 640 mobile GPU. For the original VGG-16 and ResNet-50 on ImageNet, it achieves 95ms and 48ms inference times, respectively. All these results, outperform current DNN acceleration frameworks like TensorFlow-Lite (Ten 2017) and TVM (Chen et al. 2018).

Recent work (Liu et al. 2018b) points out an interesting aspect. When one trains from scratch based on the structure (not using weight values) of a pruned model, one can often retrieve the same accuracy as the model after pruning. We incorporate this "Train-From-Scratch" process based on the results of filter pruning and combined filter and column pruning (both the best results using the enhanced SA-based search). We will observe whether accuracy can be retrieved.

Through extensive experiments, we conclude that Auto-Compress is the key to achieve ultra-high pruning rates on the number of weights and FLOPs that cannot be achieved before, while DRL cannot compete with human experts to achieve high structured pruning rates.

4.1 Results and Discussions on CIFAR-10 Dataset

Table 1 illustrates the comparison results on VGG-16 for CIFAR-10 dataset, while Table 2 shows the results on ResNet-18 (ResNet-50 for some baselines).

From the two tables we have the following conclusions. **First**, for filter/channel pruning only using manual hyperparameter determination, our method outperforms prior work 2PFPCE, NISP and AMC (both in accuracy and in pruning rate). As no other sources of improvement are exploited, this improvement is attributed to the ADMM-based algorithm equipped with purification. **Second**, the combined structured pruning outperforms filter-only pruning in both weight reduction and FLOPs reduction. For manual hyperparameter determination, the combined structured pruning enhances from 9.3× pruning rate to 44.3× in VGG-16, and enhances from 5.2× to 43.3× in ResNet-18. If we aim at the same high pruning rate for filter-only pruning, it suffers a notable accuracy drop (e.g., 88.78% accuracy at 47.4× pruning rate for VGG-16). **Third**, the enhanced SA-based hyperparameter determination outperforms DRL and manual counterparts. As can be observed in the two tables, the full AutoCompress achieves a moderate improvement in pruning rate compared with manual hyperparameter optimization, but significantly outperforms DRL-based framework (all other sources of improvement are the same). This demonstrates the statement that DRL is not compatible with ultra-high pruning rates. For relatively small pruning rates, it appears that DRL can hardly outperform manual process as well, as the improvement over 2PFPCE is less compared with the improvement over AMC.

With all sources of performance improvements effectively exploited, the full AutoCompress framework achieves 15.3× improvement in weight reduction compared with 2PFPCE and 33× improvement compared with NISP and AMC, under the same (or higher for AutoCompress) accuracy. When accounting for the different number of parameters in ResNet-18 and ResNet-50 (NISP and AMC), the improvement can be even perceived as 120×. It demonstrates the significant performance of our proposed AutoCompress framework, and also implies that the high redundancy of DNNs on CIFAR-10 dataset has not been exploited in prior work. Also the measured inference speedup on mobile GPU validates the effectiveness of the combined pruning scheme and our proposed AutoCompress framework.

Moreover, there are some interesting results on "Train-From-Scratch" cases, in response to the observations in (Liu et al. 2018b). When "Train-From-Scratch" is performed based the result of filter-only pruning, it can recover the similar accuracy. The insight is that filter/channel pruning is similar to finding a smaller DNN model. In this case, the main merit of AutoCompress framework is to discover such DNN model, especially corresponding compression rates in each layer, and our method still outperforms prior work. On the other hand, when "Train-From-Scratch" is performed based on the result of combined structured pruning, the accuracy CANNOT be recovered. This is an interesting observation. The underlying insight is that the combined pruning is not just training a smaller DNN model, but with adjustments of filter/kernel shapes. In this case, the pruned model represents a solution that cannot be achieved through DNN training only, even with detailed structures already given. In this case, weight pruning (and the AutoCompress framework) will be more valuable due to the importance of training from a full-sized DNN model.

4.2 Results and Discussions on ImageNet Dataset

In this subsection, we show the application of AutoCompress on ImageNet dataset, and more comparison results with filter-only pruning (equipped by ADMM-based core algorithm and SA-based hyperparameter determination). This will show the first source of improvement. Table 3 and Table 4 show the comparison results on VGG-16 and ResNet-18 (ResNet-50) structured pruning on ImageNet dataset, respectively. We can clearly see the advantage of AutoCompress over prior work, such as (He, Zhang, and Sun 2017) (filter pruning with manual determination), AMC (He et al. 2018) (filter pruning with DRL), and ThiNet (Luo, Wu, and Lin 2017) (filter pruning with manual determination). We can also see the advantage of AutoCompress over manual hyperparameter determination (both combined structured pruning with ADMM-based core algorithm), improving from 2.7× to 3.3× structured pruning rates on ResNet-18 (ResNet-50) under the same (Top-5) accuracy. Finally, the full AutoCompress also outperforms filter pruning only (both ADMM-based core algorithm and SA-based hyperparameter determination), improvement from 3.8× to 6.4× structured pruning rates on VGG-16 under the same (Top-5) accuracy. This demonstrates the advantage of combined filter and column pruning compared with filter pruning only, when the other sources of improvement are the same. Besides, our filter-only pruning results also outperform prior work, demonstrating the strength of proposed framework.

Table 3: Comparison results on VGG-16 for the ImageNet dataset.

Method	Top-5 Acc. Loss	Params Rt.	Objective
Filter (He, Zhang, and Sun 2017)	1.7%	≈ 4×	N/A
AMC (He et al. 2018)	1.4%	≈ 4×	N/A
Filter pruning, ADMM, SA	0.6%	3.8×	Params#
Full **AutoCompress**	0.6%	**6.4**×	Params#

Table 4: Comparison results on ResNet-18 (ResNet-50) for the ImageNet dataset.

Method	Top-5 Acc. Loss	Params Rt.	Objective
ThiNet-50 (Luo, Wu, and Lin 2017)	1.1%	≈ 2×	N/A
ThiNet-30 (Luo, Wu, and Lin 2017)	3.5%	≈ 3.3×	N/A
Filter pruning, ADMM, SA	0.8%	2.7×	Params#
Combined pruning, ADMM, manual	0.1%	2.7×	N/A
Full **AutoCompress**	0.1%	**3.3**×	Params#

Table 5: Comparison results on non-structured weight pruning on ResNet-50 using ImageNet dataset.

Method	Top-5 Acc. Loss	Params Rt.	Objective
AMC (He et al. 2018)	0%	4.8×	N/A
ADMM, manual hyper.	0%	8.0×	N/A
Full **AutoCompress**	0%	**9.2**×	Params#
Full **AutoCompress**	0.7%	**17.4**×	Params#

Last but not least, the AutoCompress framework can also be applied to non-structured pruning. For non-structured pruning on ResNet-50 model for ImageNet dataset, AutoCompress results in 9.2× non-structured pruning rate on CONV layers without accuracy loss (92.7% Top-5 accuracy), which outperforms manual hyperparameter optimization with ADMM-based pruning (8× pruning rate) and prior work AMC (4.8× pruning rate).

5 Conclusion

This work proposes AutoCompress, an automatic structured pruning framework with the following key performance improvements: (i) effectively incorporate the combination of structured pruning schemes in the automatic process; (ii) adopt the state-of-art ADMM-based structured weight pruning as the core algorithm, and propose an innovative additional purification step for further weight reduction without accuracy loss; and (iii) develop effective heuristic search method enhanced by experience-based guided search, replacing the prior deep reinforcement learning technique which has underlying incompatibility with the target pruning problem. Extensive experiments on CIFAR-10 and ImageNet datasets demonstrate that AutoCompress is the key to achieve ultra-high pruning rates on the number of weights and FLOPs that cannot be achieved before.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

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

Chen, T.; Moreau, T.; Jiang, Z.; Zheng, L.; Yan, E.; Shen, H.; Cowan, M.; Wang, L.; Hu, Y.; Ceze, L.; et al. 2018. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 578–594.

Guo, Y.; Yao, A.; and Chen, Y. 2016. Dynamic network surgery for efficient dnns. In *NIPS*, 1379–1387.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NIPS*, 1135–1143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*, 770–778.

He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; and Han, S. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, 815–832. Springer.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE ICCV*, 1389–1397.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leng, C.; Dou, Z.; Li, H.; Zhu, S.; and Jin, R. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018a. Progressive neural architecture search. In *ECCV*, 19–34.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2018b. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Luo, J.-H., and Wu, J. 2017. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE ICCV*, 5058–5066.

Min, C.; Wang, A.; Chen, Y.; Xu, W.; and Chen, X. 2018. 2pfpce: Two-phase filter pruning based on conditional entropy. *arXiv preprint arXiv:1809.02220*.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped dqn. In *NIPS*, 4026–4034.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE CVPR*, 4510–4520.

Silver, D.; Sutton, R. S.; and Müller, M. 2008. Sample-based learning and search with permanent and transient memories. In *ICML*, 968–975. ACM.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Suzuki, T. 2013. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *ICML*, 392–400.

2017. Tensorflow lite. https://www.tensorflow.org/lite.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *NIPS*, 2074–2082.

Whiteson, S.; Tanner, B.; Taylor, M. E.; and Stone, P. 2011. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 120–127. IEEE.

Ye, S.; Zhang, T.; Zhang, K.; Li, J.; Xu, K.; Yang, Y.; Yu, F.; Tang, J.; Fardad, M.; Liu, S.; et al. 2018. Progressive weight pruning of deep neural networks using admm. *arXiv preprint arXiv:1810.07378*.

Yu, R.; Li, A.; Chen, C.-F.; Lai, J.-H.; Morariu, V. I.; Han, X.; Gao, M.; Lin, C.-Y.; and Davis, L. S. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE CVPR*, 9194–9203.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, T.; Ye, S.; Zhang, K.; Tang, J.; Wen, W.; Fardad, M.; and Wang, Y. 2018a. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *ECCV*, 184–199.

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

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.