Non-Structured DNN Weight Pruning—Is It Beneficial in Any Platform?

Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Member, IEEE, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Member, IEEE, Xuehai Qian, Senior Member, IEEE, Xue Lin, Member, IEEE, Kaisheng Ma, and Yanzhi Wang

Abstract—Large deep neural network (DNN) models pose the key challenge to energy efficiency due to the significantly higher energy consumption of off-chip DRAM accesses than arithmetic or SRAM operations. It motivates the intensive research on model compression with two main approaches. Weight pruning leverages the redundancy in the number of weights and can be performed in a non-structured manner, which preserves the full matrix structure with a lower pruning rate. Weight quantization leverages the redundancy in the number of bits in weights. Compared to pruning, quantization is much more hardware-friendly and has become a “must-do” step for FPGA and ASIC implementations. Thus, any evaluation of the effectiveness of pruning should be on top of quantization. The key open question is, with quantization, what kind of pruning (non-structured versus structured) is most beneficial? This question is fundamental because the answer will determine the design aspects that we should really focus on to avoid the diminishing return of certain optimizations. This article provides a definitive answer to the question for the first time. First, we build ADMM-NN-S by extending and enhancing ADMM-NN, a recently proposed joint weight pruning and quantization framework, with the algorithmic supports for structured pruning, dynamic ADMM regulation, and masked mapping and retraining. Second, we develop a methodology for fair and fundamental comparison of non-structured and structured pruning in terms of both storage and computation efficiency. Our results show that ADMM-NN-S consistently outperforms the prior art: 1) it achieves 348×, 36×, and 8× overall weight pruning on LeNet-5, AlexNet, and ResNet-50, respectively, with (almost) zero accuracy loss and 2) we demonstrate the first fully binarized (for all layers) DNNs can be lossless in accuracy in many cases. These results provide a strong baseline and credibility of our study. Based on the proposed comparison framework, with the same accuracy and quantization, the results show that non-structured pruning is not competitive in terms of both storage and computation efficiency. Thus, we conclude that structured pruning has a greater potential compared to non-structured pruning. We encourage the community to focus on studying the DNN inference acceleration with structured sparsity.

Index Terms—Deep neural network (DNN), hardware acceleration, quantization, weight pruning.

I. INTRODUCTION

DEEPP neural networks (DNNs) with very large model sizes are the key enabler for the recent success of deep learning. However, large models incur excessive DRAM accesses which consume a significant more energy than arithmetic or SRAM operations. Thus, model compression of DNNs became an active and intensively studied research topic. These techniques, which are applied during the training phase of the DNNs, exploit the redundancy in weights. The aim is to simultaneously reduce the model size (thus, the storage requirement) and accelerate the computation for inference—all to be achieved with minor classification accuracy loss. These techniques are of particular interests to the hardware acceleration of DNN inference engine [1]–[17]. Two important model compression techniques are weight pruning and weight quantization.

Weight pruning leverages the redundancy in the number of weights. One early work [18] used heuristic and iterative weight pruning to achieve weight parameter reduction with negligible accuracy loss. It has been extended in [19]–[23] with more sophisticated heuristics. On the downside, such non-structured methods lead to irregular, sparse weight matrices (as shown in Fig. 1(a), arbitrary weight can be pruned), which rely on indices to be stored in a compressed format. As a result, they are less compatible with the data parallel execution model in GPUs and multicore CPUs. This drawback is confirmed by the throughput degradation reported in recent works [24], [25]. To overcome the limitation of non-structured pruning, recent works [24], [26] proposed the idea of incorporating regularity or “structures” in weight pruning, such as filter pruning, channel pruning, and filter shape pruning, shown...
However, each index needs to be at least 9-bit on account reduction with 3-bit quantization, maintaining 99%

in Fig. 1(b). The structured approaches maintain a full matrix with reduced dimensions, and indices are no longer needed. As a result, it leads to much higher speedups in GPUs.

Weight quantization is an orthogonal compression technique that leverages the redundancy in the number of bits of weight representation [27]–[34]. Compared to weight pruning, weight quantization is inherently more hardware-friendly, since both storage and computation of DNNs will be reduced proportionally to the weight precision without additional overhead due to indices. Moreover, multiplication operations may be eliminated with binary, ternary, or power-of-two weight quantizations [32]–[34]. Thanks to these advantages, weight quantization has been a “must-do” step for DNN inference engines. Besides FPGA and ASIC, it is also well supported in GPU, CPU, and mobile devices, e.g., [35], [36].

Given the pros and cons of non-structured/structured weight pruning and weight quantization, they need to be investigated jointly to fully understand the interactions between them. In particular, since weight quantization is a must-do step, especially for FPGA and ASIC, i.e., weight pruning will not be performed alone. The key open question is, with quantization, what kind of pruning (non-structured versus structured) is most beneficial? The answer to the question is far from obvious. Using LeNet-5 (for MNIST data set) as an example, we achieve an unprecedented 348× (non-structured) weight reduction with 3-bit quantization, maintaining 99%+ accuracy. However, each index needs to be at least 9-bit on account of 348× weight pruning. This makes index storage larger than that of weights (in addition, indices cannot be further quantized). In this example, non-structured weight pruning results in larger actual storage than structured pruning. Thus, we can see the importance of answering such question: it will determine the design aspects that we should really focus on to avoid diminishing return of certain optimizations. As shown in Fig. 2, we need answers for all platforms.

Two recent works ADMM-NN [37] and [27], that perform systematic joint weight pruning and quantization, are in the best position to perform this study. Using advanced variable-splitting optimization method ADMM (Alternating Direction Methods of Multipliers) [38]–[40], state-of-the-art results are achieved (e.g., 21× weight reduction [41] in AlexNet)—outperforming heuristic counterparts. Unfortunately, the current framework is insufficient to perform such study. First, ADMM-NN lacks the algorithmic mechanisms to enforce structured weight pruning, and guarantee the solution feasibility. Second, we lack the methodology to fairly and fundamentally compare non-structured and structured pruning in an “apple-to-apple” manner. This article is the first study to provide the answer to the open question with two key contributions.

The first contribution of this article is the development of ADMM-NN-S by extending and enhancing ADMM-NN [37]. It is extended with algorithmic supports for structured pruning. We achieve this by adjusting the constraints in each layer to express the structured requirements. For example, for filter pruning, the constraint for a layer can be specified as number of non-zero filters are less than or equal to a threshold. Moreover, we develop a systematic framework of dynamic ADMM regulation, masked mapping, and retraining to guarantee solution feasibility (satisfying all constraints) and provide high solution quality (ensuring pruning and quantization rate under the same accuracy).

The second contribution is the methodology for the fair and fundamental comparison of non-structured and structured weight pruning with quantization in place. We focus on two metrics with the same accuracy: 1) total storage (weight + indices), which is computed based on both absolute and relative indices and 2) computation efficiency, which is captured by a metrics called pruning-to-performance ratio (PPR). After pruning, suppose \( \alpha \times \) weight reduction results in \( \beta \times \) speedup, the PPR value is defined as \( \alpha/\beta \). Intuitively, the less the value of PPR, the higher the computation efficiency, the same speedup can be achieved by smaller pruning rate. For structured pruning, the PPR value is approximately one due to the absence of indices. For non-structured pruning, recent accelerators based on non-structured sparsity [37], [42]–[44] show that PPR values are larger than 2.7. We can fairly compare non-structured and structured pruning by conservatively comparing PPR: non-structured pruning is more beneficial if or higher pruning rate than structured pruning. No prior work has conducted such study and the answer to the above comparison is unknown.

The fairness of the proposed methodology is ensured due to three reasons: 1) it is performed by our new ADMM-NN-S framework that significantly outperforms prior arts (in both non-structured and structured pruning); 2) the comparison of storage and computation is hardware implementation-
agnostic; and 3) the comparison is performed at the same rate of accuracy. We also strengthen weight quantization after non-structured pruning by selectively leveraging state-of-art ternary quantization solution [45].

Based on the proposed ideas, we perform extensive and representative testing of our comparison framework with AlexNet, VGGNet, ResNet-18/50, MobileNet, and LeNet-5 models based on ImageNet, CIFAR-10, and MNIST data sets. Due to space limitation, we focus on convolutional (CONV) layers, which are the most computationally intensive layers in DNNs and are becoming the major storage as well as in state-of-art ResNet and MobileNet models. We do observe a similar (and more significant) effect on fully connected (FC) layers and on RNNs. We highlight our results and findings.

First, ADMM-NN framework guarantees solution feasibility while providing high solution quality. Our results consistently and significantly outperform prior art. This is the key to ensure the credibility of our conclusion. Specifically, we: 1) achieve unprecedented $348 \times$, $36 \times$, and $8 \times$ overall weight pruning on LeNet-5, AlexNet, and ResNet-50 models, respectively, with (almost) zero accuracy loss; 2) derive the first lossless, fully binarized (for all layers) LeNet-5 for MNIST and VGG-16 for CIFAR-10; and 3) derive the first fully binarized (for all layers) ResNet for ImageNet with reasonable accuracy loss.

Second, comparing non-structured and structured pruning, we find that the storage overhead of indices for non-structured pruning is always more than its additional weight storage reduction, thus the amount of total storage for non-structured pruning is actually larger.

In terms of computation efficiency, we find that the PPR for structured pruning in all models are less than $2.7 \times$. For the first time, our results show that, despite more flexibility and weight pruning rate, non-structured pruning is not competitive in terms of both storage and computation efficiency with quantization and the same accuracy with common sparse representation format. As a result, we reach the conclusion that our structured weight pruning framework with weight quantization is extremely suitable for current DNN inference engines as compared to non-structured weight pruning. And we recommend the community to focus on investigating to support this structured sparsity. We release codes and all the models of this work at an anonymous link: http://bit.ly/2WMQSRi.

II. MODEL COMPRESSION BACKGROUND

A. Weight Pruning

1) Non-Structured Weight Pruning: The early work by Han et al. [18] achieved $9 \times$ reduction in the number of parameters in AlexNet and $13 \times$ in VGG-16. However, most reduction is achieved in FC layers, and the $2.7 \times$ reduction achieved in CONV layers will not lead to an overall acceleration in GPUs [24]. Extensions of iterative weight pruning, such as [21] (dynamic network surgery), [19] (NeST), and [46], use more delicate algorithms such as selective weight growing and pruning. But the weight pruning rates on CONV layers are still limited, e.g., $3.1 \times$ in [21], $3.23 \times$ in [19], and $4.16 \times$ in [46] for AlexNet with no accuracy degradation. This level of non-structured weight pruning cannot guarantee sufficient speedups in GPUs. In fact, based on the enhanced ADMM-NN framework, we can achieve $11.2 \times$ non-structured weight pruning in CONV layers with almost no accuracy degradation. Ironically, it even results in $20\%$ speed degradation on an NVIDIA 1080Ti GPU.

2) Structured Weight Pruning: To overcome the limitation in non-structured, irregular weight pruning, SSL [24] proposes to learn structured sparsity at the levels of filters, channels, filter shapes, layer depth, etc. This work is among the firsts that reported the actually measured GPU accelerations. This is because CONV layers after structured pruning will transform to a full matrix multiplication with reduced matrix size. However, the weight pruning rate is limited in the prior work on structured pruning. The average weight pruning rate on CONV layers of AlexNet is only $1.4 \times$ without accuracy loss. More recently, [26] achieved $2 \times$ channel pruning with $1\%$ accuracy degradation on VGGNet. More importantly, structured pruning has never been evaluated with weight quantization.

3) Special Fine-Grained Structured Weight Pruning: Recent studies [47]–[49] investigate a new type of fine-grained structured weight pruning that combines non-structured sparsity and structured sparsity into one model. Jang et al. [47] proposed an SIMD-aware pruning method that divides weight matrix into blocks with carefully tailored size for SIMD support, and prunes weight inside each block independently. This hybrid sparsity can fully leverage the multi-level parallel hierarchy in modern CPU [50]. However, it is applicable only to devices that support SIMD operation. And another kernel-level pruning in [48] and [49] is studied, which incorporates non-structured pattern pruning inside convolution kernel and structured kernel pruning. But it needs supports from a compiler-assisted inference framework to achieve platform acceleration and is only targeted at edge computing devices such as mobile platforms.

B. Weight Quantization

Weight Quantization: This method takes advantage of the inherent redundancy in the number of bits for weight representation. Many of the prior works [27]–[34] focused on quantization of weights to binary values, ternary values, or powers of two to facilitate hardware implementation, with acceptable accuracy loss. The state-of-the-art techniques [27], [34] adopt an iterative quantization and retraining framework, with some degree of randomness incorporated into the quantization step. This method results in less than $3\%$ accuracy loss on AlexNet for binary weight quantization [27].

Compared to weight pruning, weight quantization is the major DNN model compression technique utilized in industry, due to its “hardware-friendliness” and the proportional reduction of computation and storage. Thus, weight quantization has been a must-do step in FPGA and ASIC designs of DNN inference engines. Also, it is well supported in GPUs and mobile devices, e.g., PyTorch [36] in NVIDIA GPU and TensorFlow Lite [35] for mobile devices.
C. ADMM for Weight Pruning/Quantization

Recent work [27], [37] have incorporated ADMM for DNN weight pruning and weight quantization, respectively. ADMM is a powerful tool for optimization, by decomposing an original problem into two subproblems that can be solved separately and efficiently. For example, considering optimization problem $\min_x f(x) + g(x)$. In ADMM, this problem is decomposed into two subproblems on $x$ and $z$ (auxiliary variable), which will be solved iteratively until convergence. The first subproblem derives $x$ given $z$: $\min_x f(x) + q_1(x,z)$. The second subproblem derives $z$ given $x$: $\min_z g(z) + q_2(z,x)$. Both $q_1$ and $q_2$ are quadratic functions.

ADMM is conventionally utilized to accelerate the convergence of convex optimization problems and enable distributed optimization, in which the optimality and fast convergence rate have been proven [38], [40]. As a special property, ADMM can effectively deal with a subset of combinatorial constraints and yields optimal (or at least high quality) solutions [51], [52]. Luckily, the associated constraints in the DNN weight pruning and quantization belong to this subset of combinatorial constraints, making ADMM applicable to DNN mode compression. However, due to the non-convex nature of the objective function for DNN training, there is still a lack of guarantee in the prior work [27], [37] on solution feasibility and solution quality. Moreover, [37] only supports non-structured pruning.

III. NON-STRUCTURED VERSUS STRUCTURED WEIGHT PRUNING

A. Non-Structured Pruning: Indexing Overhead

Indices are used to represent weight matrices in the sparse format, thereby achieving storage reduction in non-structured weight pruning. A representative sparse representation format is the compressed sparse row (CSR) format, which was also utilized in prior work [2], [18]. As shown in Fig. 3(a), it represents a matrix by three arrays, which, respectively, contains nonzero (weight) values, column indices, and the extents of rows. This representation requires $2n + r + 1$ numbers, where $n$ is the number of nonzero values and $r$ is the number of rows.

We call the above representation as CSR with absolute indices. Instead of storing the absolute position, we can compute the index difference and store the indices with relative position. This representation requires $2n$ numbers, where $n$ is the number of nonzero (weight) values. For further compression, one can restrict the number of bits (3 bits in this example) to represent the relative position and add a dummy zero weight when the relative position exceeds the largest value (eight for this example) that can be represented, both shown in Fig. 3(b). These cases are called CSR with relative indices.

Comparing the two options, CSR with relative indices is good for compression [18], while CSR with absolute indices leads to better hardware acceleration [42]–[44]. In this work, we aim to allow the highest freedom for non-structured pruning in storage and computation evaluations—we allow CSR with relative indices in storage calculation and CSR with absolute indices for computation estimation for non-structured pruning. There are also other sparse representation formats for sparse weight storage and corresponding hardware accelerator design [53]–[55], which provide even higher compression gains. However, we focus on the efficient execution of DNNs for the general hardware platform. To conduct fair comparisons with previous works, we choose CSR because of its better support in common deep learning packages.

B. Structured Pruning: Three Types

Wen et al. [24] introduced three types of structured pruning: filter pruning, channel pruning, and filter shape pruning, as shown in Fig. 1(b). Filter pruning removes whole filter(s); channel pruning removes whole channels; and filter shape pruning removes the weights in the same locations of all filters in one specific layer. Moreover, as shown in Fig. 4, filter pruning, and channel pruning are correlated. Pruning a filter in layer $i$ is equivalent to pruning the corresponding channel in layer $i + 1$, which is generated by this specific filter. As a result, filter pruning (and channel pruning) has a roughly quadratic effect on the weight parameter reduction (and the number of computations) of the DNNs.

The CONV operations in (one layer of) DNNs are commonly transformed to matrix multiplications by converting weight tensors and feature map tensors to matrices [13], named general matrix multiplication or GEMM, as shown in Fig. 5. From Fig. 5(b), filter pruning corresponds to reducing one
pruning has a superlinear effect on storage/computation reduc-
is in general more suitable for hardware accelerations. Thus, indices are not needed. This is why structured pruning
techniques, along with their combinations, will reduce the
multiple consecutive columns. The three structured pruning
corresponds to reducing one column and thus is also termed
row, and thus is also termed row pruning. Filter shape pruning
corresponds to reducing a filter on all layers results in over α×
reduction in the number of weight parameters. On the other
hand, column pruning has a higher degree of flexibility. These
techniques can be largely combined in order to achieve the
highest rates in reductions of computation and storage, and an
effective heuristic for the desired combination is needed.

C. Debate on Non-Structured Versus Structured: Our Stand

There are discussions on the value between structured
versus non-structured pruning. A representing work [56] stated
that the large-sparse models outperform small-dense models,
and demonstrates a significant accuracy gap between
non-structured sparsified model and structurally size-reduced
model. We found the reason for this phenomenon is the
pruning method used in [56] is heuristic, which cannot achieve
good non-structured pruning results, not to mention perform-
ing a more challenging structured pruning. In this article, the
proposed ADMM-NN-S pruning framework is systematic that solves pruning problems with an optimization technique.
When the pruning rate is low, ADMM-NN-S can even improve
accuracy consistently. In Section VII, our structured pruning
results achieve comparable high pruning rates and high
accuracy to non-structured pruning results. More importantly,
structured pruning is more suitable for hardware acceleration,
yet [56] did not discuss its results in this aspect. In this article,
we state that structured pruning is more favorable than
non-structured pruning due to its high accuracy and hardware
friendliness.

IV. ADMM-NN-S Framework

In this section, we build ADMM-NN-S, a unified solution
framework of both non-structured and structured weight
pruning, as well as weight quantization problems by extend-
ing ADMM-NN, the state-of-the-art ADMM-based framework [37]. The differences between ADMM-NN-S and
ADMM-NN are: 1) it supports structured pruning; 2) it
can guarantee solution feasibility and provide high solution
quality; and 3) we propose effective techniques for enhancing
convergence.

A. Enforcing Structured Pruning

This section discusses the extension of ADMM-NN with
structured pruning constraints. Consider an N-layer DNN
with both CONV and FC layers. The weights and biases of the
ith layer are, respectively, denoted by \( W_i \) and \( b_i \), and
the loss function associated with the DNN is denoted by
\[ f((W_i)_{i=1}^N, (b_i)_{i=1}^N) \]; see [41]. In our discussion, \( (W_i)_{i=1}^N \) and
\( (b_i)_{i=1}^N \), respectively, characterize the collection of weights and
biases from layer 1 to layer \( N \). Then DNN weight pruning or
weight quantization is formulated as optimization problem

\[
\min_{(W_i)_{i=1}^N} f((W_i)_{i=1}^N, (b_i)_{i=1}^N)
\text{s.t. } W_i \in S_i, \quad i = 1, \ldots, N.
\]  

Next, we introduce constraint sets \( S_i \)’s corresponding to the
non-structured weight pruning, different types of structured
pruning, as well as weight quantization. We use CONV layers
as an illustrative example since CONV layers are the most
computationally intensive. The problem formulation can be
well applied to FC layers [41].

The collection of weights in the ith CONV layer is a 4-D
tensor, i.e., \( W_i \in R^{A_i \times B_i \times C_i \times D_i} \), where \( A_i, B_i, C_i, \) and \( D_i \) are,
respectively, the number of filters, the number of channels in
a filter, the height of the filter, and the width of the filter,
in layer \( i \). In the following, if \( X \) denotes the weight tensor in
a specific layer, let \((X)_{a,b,c,d}\); denote the \( a \)th filter in \( X \), \((X)_{b,c,d}\);
denote the \( b \)th channel, and \((X)_{b,c,d}\) denote the collection of
weights located at position \((a, b, c, d)\) in every filter of \( X \), as
illustrated in Fig. 1(b).

1) Weight Pruning: For non-structured weight pruning, the constraint on the weights in the \( i \)th layer is \( W_i \in S_i := [X | \text{number of nonzero elements in } X \text{ is less than or equal to } a_i] \). For filter pruning (row pruning), the constraint in the
ith CONV layer becomes \( W_i \in S_i := [X | \text{the number of nonzero filters in } X \text{ is less than or equal to } \beta_i] \). For channel pruning, the constraint becomes \( W_i \in S_i := [X | \text{the number of nonzero channels in } X \text{ is less than or equal to } \gamma_i] \). Finally, for filter-shape pruning (column pruning), the constraint in the
ith CONV layer is \( W_i \in S_i := [X | \text{the number of nonzero vectors in } (X)_{b,c,d} \text{ is less than or equal to } \theta_i] \). These
\( a_i, \beta_i, \gamma_i, \) and \( \theta_i \) values are hyperparameters determined in
prior, and the determination procedure will be discussed in
Section IV-D.

2) Weight Quantization: For weight quantization, elements
in \( W_i \) assume one of \( q_{i,1}, q_{i,2}, \ldots, q_{i,M} \) values, where \( M_i \)
denotes the number of these fixed values. The \( q_{i,j} \) values

---

Fig. 5. (a) To support GEMM, weight tensor representation of a CONV layer is transformed into weight matrix representation. (b) How different structured
weight pruning schemes are implemented on weight matrix representation.
are quantization levels of weights of layer $i$ in increasing order, and we focus on equal-distance quantization (the same distance between adjacent quantization levels) to facilitate hardware implementation.

B. Enhancing Solution Feasibility and High-Solution Quality

In problem (1), the constraint is combinatorial. As a result, this problem cannot be solved directly by stochastic gradient descent methods like original DNN training. However, the form of the combinatorial constraints on $W_i$ is compatible with ADMM which is recently shown to be an effective method to deal with such clustering-like constraints [51], [52].

Despite such compatibility, it is still challenging to directly apply ADMM due to the non-convexity in the objective function. To overcome this challenge, we propose dynamic ADMM regularization, masked mapping, and retraining steps for both non-structured and structured pruning. By integrating these techniques, ADMM-NN-S can guarantee solution feasibility (satisfying all constraints) and provide high solution quality (pruning/quantization rate under the same accuracy). The procedure of ADMM-NN-S is shown in Fig. 6.

1) ADMM Regularization Step: The ADMM regularization decomposes the original problem (1) into two subproblems through: 1) defining indicator function $g_i(W_i) = 0$ if $W_i \in S_i$, corresponding to every set $S_i$; 2) incorporating auxiliary variables $Z_i$, $i = 1, \ldots, N$; and 3) adopting augmented Lagrangian [40]. These decomposed subproblems will be iteratively solved until convergence. The first subproblem is

$$
\min_{\{W_i, b_i\}} f(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N) + \sum_{i=1}^N \frac{\rho_i}{2} \|W_i - Z_i + U_i^k\|^2_F
$$

(2)

where $U_i^k := U_i^{k-1} + W_i^k - Z_i^k$. The first term in the objective function of (2) is the differentiable loss function of the DNN, and the second term is a quadratic regularization term of the $W_i$’s, which is differentiable and convex. As a result (2) can be solved by stochastic gradient descent as original DNN training. Please note that this first subproblem maintains the same form and solution for (non-structured and structured) weight pruning and quantization problems.

The second subproblem is given by

$$
\min_{\{Z_i\}} \sum_{i=1}^N g_i(Z_i) + \sum_{i=1}^N \frac{\rho_i}{2} \|W_i^k - Z_i + U_i^k\|^2_F.
$$

(3)

Note that $g_i(\cdot)$ is the indicator function of $S_i$, thus this subproblem can be solved analytically and optimally [40]. For $i = 1, \ldots, N$, the optimal solution is the Euclidean projection of $W_i^k + U_i^k$ onto $S_i$. For non-structured weight pruning, we can prove that the Euclidean projection results in keeping $\alpha_i$ elements in $W_i^{k+1} + U_i^k$ with the largest magnitudes and setting the remaining weights to zeros. For filter pruning, we first calculate $O_a = \|W_i^{k+1} + U_i^k\|^2_F$ for $a = 1, \ldots, A_i$, where $\|\cdot\|^2_F$ denotes the Frobenius norm. We then keep $\beta_i$ elements in $(W_i^{k+1} + U_i^k)_{a=1}^{A_i}$ corresponding to the $\beta_i$ largest values in $\{O_a\}_{a=1}^{A_i}$ and set the rest to zero. For channel pruning, we first calculate $O_b = \|W_i^{k+1} + U_i^k\|^2_F$ for $b = 1, \ldots, B_i$. We then keep $\gamma_i$ elements in $(W_i^{k+1} + U_i^k)_{b=1}^{B_i}$ corresponding to the $\gamma_i$ largest values in $\{O_b\}_{b=1}^{B_i}$ and set the rest to zero. The optimal solution of the second subproblem for filter shape pruning is similar, and is omitted due to space limitation. For weight quantization, we can prove that the Euclidean projection results in mapping every element of $W_i^{k+1} + U_i^k$ to the quantization level closest to that element. After both subproblems are solved, we update the dual variables $U_i$’s according to the ADMM rule [40] and thereby complete one iteration in ADMM regularization. Overall the ADMM regularization step can be understood as a smart, dynamic $L_2$ regularization, in which the regularization target $Z_i^k - U_i^k$ will change judiciously and analytically in each iteration. On the other hand, conventional regularization methods (based on $L_1$, $L_2$ norms or their combinations) use a fixed regularization target, and the penalty is applied on all the weights. This will inevitably cause accuracy degradation. Sample comparison results are in Section V.

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

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

C. Techniques for Enhancing Convergence

In this section, we discuss two techniques for enhancing convergence (rate and results): the multi-\(\rho\) method in ADMM regularization, and progressive weight pruning. We abandon the extragradient descent method in [27] as we did not find the advantage in convergence speed, not to mention the additional hyperparameters introduced by this method.

1) Increasing \(\rho\) in ADMM Regularization: The \(\rho_i\) values are the most critical hyperparameter in ADMM regularization. We start from smaller \(\rho_i\) values, say \(\rho_1 = \cdots = \rho_N = 1.5 \times 10^{-3}\), and gradually increase with ADMM iterations. This coincides with the theory of ADMM convergence [51], [52]. It in general takes 8–12 ADMM iterations for convergence, corresponding to 100–150 epochs in PyTorch. This convergence rate is comparable with the original DNN training.

2) Progressive Weight Pruning: The ADMM regularization is \(L_2\) regularization. As a result, there is a large portion of very small weight values after one round of ADMM-based (non-structured or structured) weight pruning. This gives rise to the opportunity to perform a second round of weight pruning. In practice, we perform two rounds of ADMM-based weight pruning consecutively, where the weight pruning results in the first round will be the starting point of the second round (weights that are already pruned to zero will not be recovered). This method has additional benefit of reducing the search space in each step, thereby accelerating convergence.

D. Determining Hyperparameters

Hyperparameter determination mainly refers to the determination process of pruning rate (e.g., the \(a_i\) value) and/or the number of quantization levels per layer of DNN. This is a more challenging task for pruning than quantization in general. For quantization, it is typically preferred for the same number of quantization levels for all (or most of) layers, like binarized or ternarized weights, which is preferred by hardware. For weight pruning, on the other hand, these pruning rate values are flexible and shall be judiciously determined.

As hyperparameter determination is not our primary focus, we use a heuristic method as follows. We observe that we can achieve at least \(3\times\) more weight pruning than prior, heuristic weight pruning methods without accuracy loss. Hence, we adopt the per-layer pruning rates reported in prior work and increase proportionally. In the progressive pruning procedure, we set the target of the first round to be \(1.5\times\) pruning than prior work, and the second round to be doubled based on that. We will further increase the pruning rates if there is still a margin for weight pruning without accuracy loss.

V. Non-Structured DNN Weight Pruning and Quantization Results

In this section, we demonstrate the effectiveness of ADMM-NN-S for non-structure pruning and quantization, based on ImageNet ILSVRC-2012, CIFAR-10, and MNIST data sets, using AlexNet [57], VGGNet [58], ResNet-18/ResNet-50 [59], MobileNet-V2 [60], and LeNet-5 DNN models. Due to space limitation, we only show the results of the overall DNN model (which has the most prior work for comparison), and binarized quantization of DNNs. Our implementations are based on PyTorch, and the baseline accuracy results are in many cases higher than those utilized in prior work, which reflects the recent training advances. For example, in the AlexNet model, we utilize a baseline with Top-1 accuracy 60.0% and Top-5 accuracy 82.2%, both higher than prior work (57.2% Top-1 and 80.2% Top-5). We conduct a fair comparison because we focus on relative accuracy with our baseline instead of the absolute accuracy (which has outperformed prior work).

Thanks to the compatibility of ADMM-NN-S with DNN training, directly training a DNN model using the framework achieves the same result as using a pre-trained DNN model. When a pre-trained DNN model is utilized, we limit the number of epochs in both steps in the progressive framework to be 120, similar to the original DNN training in PyTorch and is much lower than the iterative pruning heuristic [18].

A. Non-Structured Weight Pruning Results

1) AlexNet Results for ImageNet Data Set: Table I compares the overall pruning rates of the whole AlexNet model (CONV and FC layers) versus accuracy, between the proposed framework and various prior methods. We can clearly observe that the proposed framework outperforms prior methods, including the prior ADMM method [41]. With almost no accuracy loss even based on the high baseline accuracy, we achieve a \(36\times\) overall pruning rate. We achieve a notable \(63\times\) weight reduction with 80.3% Top-5 accuracy, just slightly below the baseline accuracy in prior work.

Fig. 7 illustrates the absolute top-5 accuracy for different pruning methods, on AlexNet model for ImageNet data set. These methods include our proposed solution, iterative pruning [18], fixed regularization techniques like \(L_1\) and \(L_2\) regularizations, and projected gradient descent (PGD). The results clearly show that the proposed method outperforms the others both in absolute accuracy and in relative accuracy loss.

2) ResNet-50 Results for ImageNet Data Set: Due to the lack of existing effective pruning results, we conduct uniform weight pruning—use the same pruning rate for all CONV and FC layers. The results are shown in Table II. We achieve an \(8\times\) overall pruning rate (also \(8\times\) pruning rate on CONV layers)
Fig. 7. Top-5 accuracies for different pruning methods on AlexNet for ImageNet data set.

**TABLE II**
Comparisons of Overall Weight Pruning Results on ResNet-50 for ImageNet Data Set

| Method               | Top-5 Acc. Loss | Pruning rate |
|----------------------|-----------------|--------------|
| Uncompressed         | 0.0%            | 1×           |
| Fine-grained [46]    | 0.1%            | 2.6×         |
| ADMM-NN [37]         | 0.0%            | 7×           |
| Our method           | 0.0%            | 8×           |
| Our method           | 0.7%            | 17.4×        |

**TABLE III**
Comparisons of Overall Weight Pruning Results on MobileNet-V2 for ImageNet Data Set

| Method               | Top-5 Acc. Loss | Pruning rate |
|----------------------|-----------------|--------------|
| Uncompressed         | 0.0%            | 1×           |
| Our method           | 0.6%            | 2.0×         |

**TABLE IV**
Our Weight Pruning Results on MobileNet-V2 for CIFAR-10 Data Set

| Method               | Accuracy  | Pruning rate |
|----------------------|-----------|--------------|
| Uncompressed         | 95.07%    | 1×           |
| Our method           | 94.95%    | 5.7×         |
| Our method           | 94.70%    | 6.7×         |
| Our method           | 93.75%    | 10×          |

on ResNet-50 without accuracy loss. These results clearly outperform the prior work.

3) MobileNet-V2 Results for ImageNet Data Set: Table III shows the non-structured pruning results on MobileNet-V2 for ImageNet data set. MobileNet-V2 is a lightweight network structure with less than 2.2M parameters. The Top-5 accuracy of the uncompressed model is 90.4%. Due to the fact that MobileNet-V2 is one of the most difficult networks to prune, there are no existing effective irregular pruning results. Thus, we only include our results in Table III.

4) MobileNet-V2 Results for CIFAR-10 Data Set: The baseline accuracy is as high as 95.07% due to the adoption of the mixup technique. We present our results in Table IV due to the lack of prior work for a fair comparison. We achieve 5.7× weight pruning with almost no accuracy loss, starting from the high-accuracy baseline. We achieve 10× weight pruning (which is highly challenging for MobileNet) with only 1.3% accuracy loss.

5) LeNet-5 Results for MNIST Data Set: Table V demonstrates the comparison results on LeNet-5 model using MNIST data set. We achieve an unprecedented 348× overall weight reduction with almost no accuracy loss. It clearly outperforms prior methods including the one-shot ADMM-based method [41].

B. Binary Weight Quantization Results

Due to space limitation, we mainly show the results on fully binarized DNN models (i.e., weights in all layers, including the first and the last, are binarized), which is a highly challenging task. Please note that the amount of prior work on fully binarized weight quantization is very limited due to the highly challenging nature.

1) Weight Quantization Results on LeNet-5 and CIFAR-10: To the best of our knowledge, we achieve the first lossless, fully binarized LeNet-5 model. The accuracy is still 99.21%, lossless compared with baseline. In prior works, achieving lossless is challenging even for MNIST. For example, recent work [61] results in 2.3% accuracy degradation on MNIST for full binarization, with baseline accuracy 98.66%. We also achieve the first lossless, fully binarized VGG-16 for CIFAR-10. The accuracy is 93.53%. We would like to point out that fully ternarized quantization results in 93.66% accuracy. Table VI shows our results and comparisons.

2) Binary Weight Quantization Results on ResNet for ImageNet: The binarization of ResNet models on ImageNet data set is widely acknowledged as an extremely challenging task. As a result, there is very limited prior work (e.g., the prior ADMM-based method [27]) with binarization results on ResNet models. As [27] targets ResNet-18, we make a fair comparison on the same model. Table VII demonstrates the comparison results (Top-5 accuracy loss). In prior work, by default, the first and last layers are not quantized (to 8 bits) as these layers have a significant effect on overall accuracy. When leaving the first and last layers unquantized, we observe higher accuracy compared with the prior method. The Top-1 accuracy has similar result: 3.8% degradation in our method and 4.3% in [27].
Furthermore, we can derive a fully binarized ResNet-18, in which weights in all layers are binarized. The accuracy degradation is 5.8%, which is noticeable and shows that the full binarization of ResNet is a challenging task even for the proposed framework. We did not find prior work to compare with this result.

3) Summary: The results presented in this section show that ADMM-NN-S can achieve better results compared to state-of-the-art. In certain cases, ADMM-NN-S achieves unprecedented weight reduction. These results provide a strong baseline and credibility of our study.

VI. NON-STRUCTURED VERSUS STRUCTURED: THE COMPARISONS METHODOLOGY

A. Motivation Example

The Section V has shown the superior results on joint weight pruning and quantization. Using LeNet-5 (MNIST data set) as an example, we achieve an unprecedented $348 \times$ non-structured weight reduction together with 3-bit quantization, maintaining 99%+ accuracy. When indices are not accounted for, the overall compression rate is an unprecedented $3712 \times$ compared with the original LeNet-5 model without compression. However, each index needs to be at least 9-bit considering $348 \times$ weight pruning. This makes index storage even larger than weights, and indices cannot be further quantized. As a result, non-structured weight pruning in fact results in larger actual storage than structured pruning.

The fundamental phenomena shown here is that, with quantization the weight reduction by non-structured pruning is offset by the extra index storage. It motivates us to study whether it is a common trend with weight quantization in place? If the answer is yes, then the value of non-structured weight pruning will be further in doubt. This is because non-structured pruning is already less preferred for GPU and CPUs [24], [25], the only benefit is the potentially higher pruning rates due to greater pruning flexibility. If this benefit is also lost, there will be nearly no merit of non-structured sparsity for hardware acceleration of DNNs, considering the impacts on computation efficiency and degraded parallelism. Importantly, such a conclusion will also be true for FPGA and ASIC and guide us to the design aspects that we should really focus on.

In this section, we conduct the first(to the best of our knowledge) comprehensive study to understand the value of non-structured and structured pruning, with quantization in place and the same accuracy. It is worth noting that without ADMM-NN-S framework, this study is not possible—we need a framework that achieves competitive results and can jointly perform both weight pruning and quantization.

B. Hardware Implementation-Agnostic Comparison Methodology

We conduct a fair comparison between non-structured and structured weight pruning with quantization in place, based on the unified solution framework. Note that the comparison framework is more FPGA and ASIC oriented as flexible weight quantization is assumed. However, we would like to point out that a moderate, fixed weight quantization, e.g., 8 bit, supported in GPU [36], TPU [62], and mobile devices [35], will result in a similar conclusion.

The key characteristic of our comparison framework is that it is hardware implementation-agnostic. Our intention is that the comparison results will be independent of specific hardware implementations, and as a result, the conclusion will unlikely to change for architectural advances in either type of pruning. Therefore, we directly compare the amounts of storage and estimated computation efficiency for non-structured and structured weight pruning with quantization in place, which captures the fundamental tradeoffs. Intuitively, storage is measured as the total weight and index storage with quantization in place. Storage of intermediate results is not considered, and this favors non-structured pruning—structured, filter/channel pruning will likely benefit more in intermediate results storage reduction.

On the other hand, computation efficiency is estimated using the PPR values derived from prior work on non-structured sparsity accelerators [37], [42]–[44]. For structured pruning, $\alpha \times$ weight reduction results in around $\alpha \times$ speedup (slightly higher or lower depending on platform and problem), and the PPR value is approximately 1. For non-structured pruning, $\alpha \times$ weight reduction only results in $\beta \times$ speedup with $\beta < \alpha$. In the state-of-art tapeouts [42], the PPR value $\alpha/\beta > 3$, which is close to three with a low pruning rate and higher than four for a high pruning rate. In synthesis results [37], [43], [44], this PPR value ranges from 2.7 to 3.5. We use the smallest value 2.7 that favors non-structured pruning the most. In other words, if non-structured pruning achieves more than $2.7 \times$ pruning rate than structured one (or equivalently, structured pruning rate is less than 37% of non-structured one) under the same accuracy and quantization level, the former is more preferred in terms of computation. Otherwise, the latter is more preferred.

C. Maintaining the Same Accuracy for Comparison

The proposed comparison is performed under the same accuracy for non-structured and structured pruning with quantization in place. The precise accuracy control, which is challenging for prior work, is enabled by the unified solution framework. For most cases, we would like to have (almost) no accuracy degradation compared with the baseline DNN model without pruning or quantization. For non-structured pruning, it is achieved in two steps: 1) perform weight pruning to the maximum extent such that there will be no accuracy loss and 2) perform weight quantization (hopefully) not to cause accuracy loss. For structured pruning, we give priority to column pruning, and perform three steps: 1) perform column pruning to the maximum extent without accuracy loss; 2)
perform filter pruning and reduce corresponding redundant channels; and 3) perform weight quantization (hopefully) without accuracy loss. Fig. 8 illustrates the procedure for maintaining accuracy. Of course, the proposed framework is also applicable if certain accuracy degradation is allowed. A larger margin of accuracy loss in general favors structured pruning, because higher pruning rates can be achieved for both pruning schemes, but non-structured pruning requires more bits for indices.

There is more subtlety in the combination of non-structured pruning and quantization. If weight is non-zero after pruning but quantized to zero, this weight can be added to the pruned list to achieve a higher pruning rate. Note that this phenomenon does not apply to structured pruning. To better exploit this phenomenon and achieve even higher storage/computation reduction for non-structured pruning (plus quantization), we leverage the state-of-the-art ternary quantization technique [45] with dedicated optimizations. We apply this technique for weight quantization after non-structured pruning in cases when it outperforms our proposed method, thereby providing enough opportunity for non-structured weight pruning.

### VII. Comparison of Non-Structured and Structured Weight Pruning

Due to space limitation, we focus on CONV layers, which are the most computationally intensive layers in DNNs and are becoming the major storage in the state-of-art ResNet and MobileNet models. We do observe a similar (and more significant) effect on FC layers and on RNNs, without providing detailed results due to space.

As discussed in Section V, our implementations are based on PyTorch with high baseline accuracies. We limit the number of epochs in both structured pruning and non-structured pruning to be 240 (much lower than the iterative pruning heuristic [18]), and the number of epochs in weight quantization to be 120. We adopt hyperparameter determination heuristic discussed in Section IV-D for both structured and non-structured pruning.

For non-structured weight pruning, we show results on both CSR with relative indices and with absolute indices. The former is more appropriate for storage reduction, but the latter achieves higher computation efficiency. For absolute indices, we assume $4K = 64 	imes 64$ blocks that are reasonable for hardware [42]. Besides the comparison between two pruning schemes, our results also consistently outperform prior work, in terms of both non-structured and structured pruning, as well as a combination with weight quantization.

#### A. Comparison Results on ImageNet Data Set

Tables VIII–X demonstrate the comparison results using AlexNet, ResNet-18 (ResNet-50 in prior work NISP and THINet), and ImageNet data set. In these tables, “CONV Prune Rate” refers to the reduction ratio in the number of weights in overall CONV layers, and the number of remaining weights is “CONV No. of Weights.”

#### TABLE VIII

| Method | Top-5 Accuracy (%) | CONV Prune Rate (%) | CONV No. of Weights | CONV Quant Bits | CONV Weight Store | Index Bits | Weight/Index Store (Relative) | Weight/Index Store (Absolute) | CONV Compress Rate |
|--------|-------------------|---------------------|--------------------|----------------|------------------|-----------|------------------------------|------------------------|------------------|
| Baseline AlexNet | 82.2% | 1.0x | 2.3M | 32 | 0.85MB | 4 | 9.3MB | 9.3MB | 1.0x |
| Non-structured | | | | | | | | | |
| Han [63] | 80.5% | 2.7x | 0.59M | 8 | 0.5MB | 4 | 1.5MB | N/A | N/A |
| Dynamic surg. [21] | 80.0% | 1.2x | 0.74M | N/A | N/A | N/A | N/A | N/A |
| Neural [19] | 80.3% | 3.2x | 0.71M | N/A | N/A | N/A | N/A | N/A |
| Fine-grained [46] | 80.3% | 4.1x | 0.55M | N/A | N/A | N/A | N/A | N/A |
| ours’ | 81.9% | 11.2x | 0.3M | 7 | 0.26MB | 6 | 0.63MB | 0.63MB | 25.2x |
| Structured | | | | | | | | | |
| SSL [34] | 80.4% | 1.4x | 1.6M | N/A | N/A | - | N/A | N/A | N/A |
| Taylor [64] | 79.8% | 2.5x | 0.93M | N/A | N/A | - | N/A | N/A | N/A |
| NDP [55] | 80.2% | 1.9x | 1.2M | N/A | N/A | - | N/A | N/A | N/A |
| ours’ | 81.8% | 5.1x | 0.65M | 7 | 0.56MB | - | 0.56MB | 0.56MB | 23.3x |

#### TABLE IX

| Method | Top-5 Accuracy (%) | CONV Prune Rate (%) | CONV No. of Weights | CONV Quant Bits | CONV Weight Store | Index Bits | Weight/Index Store (Relative) | Weight/Index Store (Absolute) | CONV Compress Rate |
|--------|-------------------|---------------------|--------------------|----------------|------------------|-----------|------------------------------|------------------------|------------------|
| Baseline ResNet-18 | 89.1% | 1.0x | 11.2M | 32 | 44.75MB | - | 44.75MB | 44.75MB | 1.0x |
| Non-structured | | | | | | | | | |
| ours’ | 89.1% | 6.4x | 1.75M | 6 | 1.33MB | 5 | 2.97MB | N/A | N/A |
| Structured | | | | | | | | | |
| DCP [66] | 87.6% | 2.0x | 3.75M | N/A | N/A | - | N/A | N/A | N/A |
| DCP [66] | 85.7% | 3.3x | 3.5M | N/A | N/A | - | N/A | N/A | N/A |
| ThINet-50 [67] | 90.7% | 2.0x | 12.8M | N/A | N/A | - | N/A | N/A | N/A |
| ThINet-50 [67] | 88.3% | 3.3x | 7.7M | N/A | N/A | - | N/A | N/A | N/A |
| NIP [65] | 90.2% | 1.8x | 14.2M | N/A | N/A | - | N/A | N/A | N/A |
| ours’ | 89.1% | 2.5x | 4.64M | 6 | 3.34MB | 5 | 3.34MB | 3.34MB | 13.4x |
| ours’ | 87.8% | 4.3x | 2.60M | 6 | 1.95MB | - | 1.95MB | 1.95MB | 22.9x |

Fig. 8. Procedure for maintaining accuracy.
Weights.” “CONV Quant Bits” refers to the number of bits used for equal-distance weight quantization while “CONV Weight Store” is the storage required only for weights (not account for indices). “Index Bits” refers to the number of bits in CSR with relative indices. In our results, we already optimized this index bit value to minimize the overall storage (accounting for the additional dummy zeros as well). The next two columns refer to the total storage size accounting for relative indices and absolute indices, respectively. For structured pruning, they are the same as weight storage. The final column “CONV Compress Rate” refers to the storage compression rate compared with the original baseline DNN model without compression, assuming relative indices that are more favorable to non-structured pruning. We use “N/A” if the specific prior work only focuses on weight pruning without performing quantization.

It can be observed that we achieve significant pruning rate gains for both non-structured and structured pruning. Especially, for structured pruning, we achieve 5.1× and 2.5× structured weight pruning in CONV layers of AlexNet and ResNet-18 models, respectively, without accuracy loss. We further achieve 4.3× structured pruning with minor accuracy loss of around 1%. For ResNet on ImageNet data set, it is difficult for prior work to achieve lossless structured pruning. For example, [26] results in 1% accuracy loss with 2× structured pruning, on ResNet-50 model with more redundancy. For MobileNet-V2 on the ImageNet data set, the uncompressed model training is already very challenging, which implies pruning such a network is even harder. But we still achieve a 1.7× structured pruning rate with only 0.7% accuracy loss.

When comparing non-structured versus structured pruning, the overall CONV compression rate is comparable for the AlexNet case and the 1% accuracy loss case for ResNet-18. For the lossless case in ResNet-18, non-structured pruning is slightly better in storage, especially when relative indices are utilized. This is because the number of bits for indexing is relatively small in this case, and the slight benefit will diminish if a certain accuracy loss is tolerable. The occasional gain cannot outweigh the difficulty in hardware support of non-structured sparsity. It would be difficult to choose non-structured pruning over the other one even if the storage results are comparable.

### Comparison Results on CIFAR-10 Data Set

Tables XI and XII demonstrate the comparison results using VGG-16 and ResNet-18 models on CIFAR-10 data set.
We observe that very significant pruning rates can be achieved compared with prior work (over 35× improvement in certain cases). We investigated deeper and found that the underlying reason is the CIFAR-10 data set itself, in that it is both “simple” and “difficult.” “Simple” means that the input image scale is small and the number of classes is only ten; while “difficult” means that input images are blurred and feature extraction is not straightforward. As a result, researchers tend to migrate large-scale DNN models originally designed for ImageNet, such as VGG-16 and ResNet-18 (prior work even used ResNet-50). Consequently, there is a significant margin of model compression, which can be exploited in the proposed systematic framework but difficult for heuristic methods.

Another observation is that non-structured pruning has only marginal gain in pruning rates (reduction in the number of weights) compared with structured ones. Our hypothesis is that it is due to the high search space in non-structured pruning. Together with a large number of index bits due to high pruning rates, non-structured pruning is not preferable compared with structured ones considering total storage size. The storage size gap is becoming surprisingly large when absolute indices are considered. Together with a large number of index bits due to high pruning rates, non-structured pruning is not preferable compared with structured ones considering total storage size. The storage size gap is becoming surprisingly large when absolute indices are considered.

C. Comparison Results on MNIST Data Set

Table XIV demonstrates the comparison results using MobileNet-V2 model on CIFAR-10 data set. MobileNet is already compact and relatively difficult for further weight pruning, but we still achieve 5× structured pruning along with 4-bit quantization. Again non-structured pruning only shows a minor gain in weight reduction, and it is not preferable considering indexing overheads.

D. Comparison on Computation Efficiency

We have shown that non-structured pruning is not preferable in terms of storage even assuming the storage-friendly CSR format with relative indices, not to mention absolute indices. Based on our methodology, we find that computation efficiency shows a similar trend.

As discussed before, structured pruning will have higher computation efficiency if it achieves more than 37% in the pruning rate as non-structured pruning. In all our testing, the ratio between weight pruning rates of structured versus non-structured pruning ranges from 40% to 87%, with a large variation but consistently higher than 37%. Even for the 40% case, the choice is clear considering the difficulty in hardware design for non-structured sparsity. As a result, we draw a conclusion that non-structured weight pruning is in general not preferred compared with structured pruning across different platforms, application scenarios, DNN types, etc.

VIII. Conclusion

Non-structured and structured weight pruning and weight quantization are major methods for model compression, but the interaction among different techniques are never clearly understood. This article is the first to investigate the value of non-structured and structured DNN weight pruning, when the weight quantization is in place. We build ADMM-NN-S, a joint weight pruning and quantization framework with algorithmic supports for structured pruning, dynamic ADMM regulation, and masked mapping and retraining. To perform a fair and fundamental comparison between non-structured and structured pruning in an implementation-agnostic manner, we propose a methodology that captures storage overhead and computation efficiency. We perform extensive and representative testing of ADMM-NN-S with AlexNet, VGGNet, ResNet-18/50, MobileNet, and LeNet-5 models based on ImageNet, CIFAR-10, and MNIST data sets. We show that ADMM-NN-S can significant outperform the state-of-the-art results for non-structured pruning with quantization. More importantly, for the first time, we show that with quantization in place and the same accuracy, structured pruning is more preferable to non-structured pruning in terms of computation efficiency.

REFERENCES

[1] Y. Li et al., “A network-centric hardware/algorithm co-design to accelerate distributed training of deep neural networks,” in Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO), Oct. 2018, pp. 175–188.

[2] S. Han et al., “EIE: Efficient inference engine on compressed deep neural network,” in Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2016, pp. 243–254.

[3] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, “EVA²: Exploiting temporal redundancy in live computer vision,” in Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2018, pp. 533–546.

[4] K. Hegde, J. Yu, R. Agrawal, M. Yan, M. Pellauer, and C. Fletcher, “UCNN: Exploiting computational reuse in deep neural networks via weight repetition,” in Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2018, pp. 674–687.

[5] A. Ren et al., “SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing,” ACM SIGOPS Oper. Syst. Rev., vol. 51, no. 2, pp. 405–418, Apr. 2017.
[56] M. Zhu and S. Gupta, “To prune, or not to prune: Exploring the efficacy of pruning for model compression,” 2017, arXiv:1710.01878. [Online]. Available: http://arxiv.org/abs/1710.01878

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[58] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556

[59] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[60] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4510–4520.

[61] X. Zhu, W. Zhou, and H. Li, “Adaptive layerwise quantization for deep neural network compression,” in Proc. IEEE Int. Conf. Multimedia Expo (ICME), Jul. 2018, pp. 1–6.

[62] Google Supercharges Machine Learning Tasks With TPU Custom Chip. Accessed: Jan. 2019. [Online]. Available: https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html

[63] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2016.

[64] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” 2016, arXiv:1611.06440. [Online]. Available: http://arxiv.org/abs/1611.06440

[65] R. Yu et al., “NISP: Pruning networks using neuron importance score propagation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 9194–9203.

[66] Z. Zhuang et al., “Discrimination-aware channel pruning for deep neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 875–886.

[67] J.-H. Luo, J. Wu, and W. Lin, “ThiNet: A filter level pruning method for deep neural network compression,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5058–5066.

[68] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: Automl for model compression and acceleration on mobile devices,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 784–800.

[69] E. Elsen, M. Dukhan, T. Gale, and K. Simonyan, “Fast sparse ConvNets,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 14629–14638.

[70] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the value of network pruning,” in Proc. ICLR, 2019, pp. 1–21.

[71] C. Min, A. Wang, Y. Chen, W. Xu, and X. Chen, “2FPFCE: Two-phase filter pruning based on conditional entropy,” 2018, arXiv:1809.02220. [Online]. Available: http://arxiv.org/abs/1809.02220

[72] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” in Proc. ICLR, 2017, pp. 1–13.

Shaokai Ye received the B.S. degree in computer engineering from Saint Louis University, St. Louis, MO, USA, in 2015, and the M.S. degree in computer engineering from Syracuse University, Syracuse, NY, USA, in 2018, under the supervision of Dr. Yanzhi Wang. He is currently pursuing the Ph.D. degree with the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, under the supervision of Dr. Mackenzie Mathis. His research interest includes making AI more efficient and robust.

Zhezhi He (Member, IEEE) received the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2020. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His primary research interests include neuromorphic computing, efficient and secure deep learning, in-memory computing, and electronic design automation.

Linfeng Zhang is currently pursuing the Ph.D. degree with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China, supervised by Kaisheng Ma. He publishes papers on conferences, including the Conference on Neural Information Processing Systems (NeurIPS), the International Conference on Learning Representations (ICLR), the Conference on Computer Vision and Pattern Recognition (CVPR), and the International Conference on Computer Vision (ICCV). His research interests include computer vision, and deep neural network compression and acceleration.

Geng Yuan received the M.S. degree in electrical engineering from Syracuse University, Syracuse, NY, USA, in 2016. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA, supervised by Dr. Yanzhi Wang. His research interests include high-performance and energy-efficient deep learning system design, computer vision, and deep learning architecture design with emerging technologies.

Sia Huat Tan was born in Sarawak, Malaysia. He received the B.S. degree (Hons.) in computer science and information engineering from National Chung Hsing University, Taian, Taiwan, in 2019. He is currently pursuing the M.S. degree with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China. He is currently working on the design of neural network models to improve the efficiency and robustness against common corruptions and adversarial examples. His research interest includes computer vision.
Zhengang Li received the B.E. degree in electronic information engineering from Zhejiang University, Zhejiang, China, in 2017. He is currently pursuing the Ph.D. degree in computer engineering with Northeastern University, Boston, MA, USA, under Prof. Yanzhi Wang.

His current research interests include model compression of deep neural networks, machine learning algorithm, and computer vision.

Deliang Fan (Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2012 and 2015, respectively.

He is currently an Assistant Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. Before joining ASU in 2019, he was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA. His primary research interests include energy-efficient and high-performance big data processing-in-memory circuit, architecture, and algorithm, with applications in deep neural networks, data encryption, graph processing, and bioinformatics acceleration-in-memory systems; hardware-aware deep learning optimization; brain-inspired (neuromorphic) computing; and AI security. He has authored or coauthored more than 110 peer-reviewed international journal/conference papers in above areas.

Dr. Fan was a recipient of the Best Paper Award of 2019 ACM Great Lakes Symposium on VLSI (GLSVLSI), the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), and the 2017 IEEE ISVLSI. He is also the Technical Area Chair of Design Automation Conference (DAC) 2021, GLSVLSI 2019/2020/2021, and the International Symposium on Quality Electronic Design (ISQED) 2019/2020/2021; and the Financial Chair of ISVLSI 2021. Please refer to https://dfan.engineering.asu.edu/ for more details.

Xuehai Qian (Senior Member, IEEE) received the B.Sc. degree from Beihang University, Beijing, China, in 2004, the M.Sc. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2007, and the Ph.D. degree from the Computer Science Department, University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2013.

He is currently an Assistant Professor with the Ming Hsieh Department of Electrical Engineering and the Department of Computer Science, University of Southern California, Los Angeles, CA, USA. His research interests include computer architecture, architectural support for programming productivity, and correctness of parallel programs.

Xue Lin (Member, IEEE) received the bachelor’s degree in microelectronics from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA, in 2016.

She has been an Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA, since 2017. Her research interests include deep learning security and hardware acceleration, machine learning and computing in cyber-physical systems, high-performance and mobile cloud computing systems, and VLSI.

Kaisheng Ma received the Ph.D. degree in computer science and engineering from Pennsylvania State University, State College, PA, USA, in 2018, following Prof. Vijaykrishnan Naranayan at Penn State, Yuan Xie at the University of California, Santa Barbara (UCSB), Santa Barbara, CA, USA, and Jack Sampson at Penn State.

He is currently a tenure-track Assistant Professor of computer science with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China. He leads the Algorithms, Architecture and Chipset Laboratory (ArChip), Tsinghua University. His research interests include interdisciplinary fields of human–machine interface (HMI), brain-inspired AI algorithm design, computer vision for self-driving, compact model design, computer architecture, and microelectronics. He believes in vertical integration for systems, with a special emphasis on implanted medical and neural devices, and chip level system solutions for self-driving systems.

Dr. Ma’s previous research on Nonvolatile Processor and Energy Harvesting won the 2018 European Design and Automation Association (EDAA) Best Dissertation Award. He publishes papers on conferences, including NeuralIPS, the International Conference on Computer Vision (ICCV), Association for the Advancement of Artificial Intelligence (AAAI), the Conference on Computer Vision and Pattern Recognition (CVPR), International Symposium on Computer Architecture (ISCA), International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLoS), International Symposium on Microarchitecture (MICRO), International Symposium on High-Performance Computer Architecture (HPCA), and the Design Automation Conference (DAC). He has won many awards, including the 2015 HPCA Best Paper Award, the 2016 IEEE MICRO Top Picks, the 2017 ASP-DAC Best Paper Award, and the 2018 EDAA Best Dissertation Award. He has many honors, including the 2016 Penn State CSE Department Best Graduate Research, the 2016 Cover Feature of NSF ASSIST Engineering Research Scholarship, and the Yang Fuqing and WANG Yangyu Scholarships.

Yanzhi Wang received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA, in 2014.

His research interests include energy-efficient and high-performance implementations of deep learning and artificial intelligence systems, emerging deep learning algorithms/systems, generative adversarial networks, and deep reinforcement learning.