Alternating Direction Method of Multipliers for Quantization

Tianjian Huang* tianjian@usc.edu
Prajwal Singhania† prajwal1210@gmail.com
Maziar Sanjabi‡ maziars@fb.com
Pabitra Mitra† pabitra@gmail.com
Meisam Razaviyayn* razaviya@usc.edu

*University of Southern California
†Indian Institute of Technology Kharagpur
‡Facebook AI

Abstract

Quantization of the parameters of machine learning models, such as deep neural networks, requires solving constrained optimization problems, where the constraint set is formed by the Cartesian product of many simple discrete sets. For such optimization problems, we study the performance of the Alternating Direction Method of Multipliers for Quantization (`ADMM-Q`) algorithm, which is a variant of the widely-used ADMM method applied to our discrete optimization problem. We establish the convergence of the iterates of `ADMM-Q` to certain *stationary points*. In addition, our results shows that the Lagrangian function of ADMM converges monotonically. To the best of our knowledge, this is the first analysis of an ADMM-type method for problems with discrete variables/constraints. Based on our theoretical insights, we develop a few variants of `ADMM-Q` that can handle inexact update rules, and have improved performance via the use of "soft projection" and "injecting randomness" to the algorithm. We empirically evaluate the efficacy of our proposed approaches on two problem: 1) solving quantized quadratic optimization problems and 2) training neural networks. Our numerical experiments shows that `ADMM-Q` outperforms other competing algorithms.

Proceedings of the 24$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

1 Introduction

The fields of machine learning and artificial intelligence have experienced significant advancements in recent years. Despite this rapid growth, the extreme energy consumption of many existing machine learning models prevents their use in low-power devices. As a solution, *quantized* and *binarized* training of these models have been proposed in recent years Courbariaux et al. [2015, 2016], Rastegari et al. [2016], Szegedy et al. [2013]. This procedure requires training a machine learning model that has low training/test error, and at the same time, low power/storage requirement. More precisely, the parameters of the machine learning model must lie in a discrete set (e.g. the weights of the neural network should be binary). The goal is to improve the energy and storage efficiency of the model by simplifying the required storage/computation in the inference phase.

To obtain accurate quantized models, a wide range of training techniques have been proposed. Among them, Alternating Direction Method of Multipliers (ADMM) has recently gained popularity and resulted in training highly accurate machine learning models with super low power consumption Ye et al. [2018, 2019], Yuan et al. [2019b,a], Leng et al. [2018], Lin et al. [2019], Zhang et al. [2018], Liu et al. [2020], Ren et al. [2019], Li et al. [2019]. Despite this empirical success, the theoretical understanding of ADMM for solving discrete optimization problems, such as training binarized neural networks, is almost nonexistent. As a first step toward better understanding the behavior of this algorithm in solving discrete problems, in this paper we aim at *studying the behavior of the ADMM algorithm in discrete optimization through answering the following simple yet fundamental questions*:

Brief Problem Description. Assume ADMM algorithm is applied to a nonconvex discrete optimization problem such as training binarized/quantized neural networks.

- Is ADMM guaranteed to improve the objective function over the iterates? Can we end up at a point that is worse than the initial point?
- What can we say about the "limit points" of the iterates generated by the ADMM algorithm in this discrete context?
- Can ADMM tolerate inexact, randomized, or stochastic computations?
- Is ADMM better than simple algorithms such as projected gradient descent when applied to this discrete problem?

The answer to the above fundamental questions is non-trivial. This lack of understanding is due to the non-monotonic behavior of the objective function through ADMM iterates as well as the highly fragile relations between the primal and dual variables in this discrete optimization setting. In this paper, we (partially) answer the above questions by first showing that the `ADMM-Q` algorithm, which is a variant of ADMM in discrete setting, indeed improves the objective over iterations. We analyze the limit points of the iterates generated by `ADMM-Q` and show that every limit point of the iterates satisfies certain *stationarity property*. Then, we extend our analysis to inexact and randomized update rules that happen in many practical problems such as training binarized neural networks. Finally, we evaluate the performance of `ADMM-Q` and its extensions in our numerical experiments. The goal of our numerical experiments is not to obtain the best performance in a particular application or existing benchmark problems, but instead to better understand the behavior of `ADMM-Q` method. Notice that `ADMM-Q` has already been used in other papers and its efficiency (combined with other training heuristics) has been established in the literature for different problems Leng et al. [2018], Lin et al. [2019], Yuan et al. [2019b,a]. Moreover, to obtain a better understanding of `ADMM-Q`, we avoid using heuristics such as Straight-through Estimators, scaling factor, not binarizing last layer, playing with the architecture, which has been used in other papers Bengio et al. [2013], Rastegari et al. [2016], Darabi et al. [2018], Tang et al. [2017]. While these heuristics (combined with exact tuning of many parameters) can significantly improve the performance of the method, they make the scientific study of the core `ADMM-Q` algorithm almost impossible by bringing a lot of other not well-understood approaches to the table. Thus, in our numerical experiments, instead of aiming for the best possible performance, obtained by using multiple heuristics, we only focus on the empirical performance of the core quantization algorithm.

1.1 State of the art

This paper studies the behavior of the ADMM algorithm when applied to nonconvex discrete optimization problems. This is closely tied to the previous studies on the ADMM algorithm and training quantized machine learning models. Here we briefly review some of the existing works in each of these two categories:

Quantized machine learning models. In recent years, there have been numerous works on the quantization of machine learning models—specifically neural networks. One of the first works towards this was BinaryConnect Courbariaux et al. [2015] which used the "Straight Through Estimator" (STE) Bengio et al. [2013] to provide a "from-scratch" training method with binary weights. BinaryNet Courbariaux et al. [2016] extended upon this idea to binarize both weights and activations, replacing complex convolutions with simpler bit-wise operations and significantly reducing the computational complexity. These works performed very well on smaller datasets like MNIST, SVHN and CIFAR-10, and provided an important direction for compression of neural networks. However, their performance on ImageNet Deng et al. [2009] classification was poor. XNOR-Net Rastegari et al. [2016] was one of the first works to improve binarized CNNs for ImageNet classification by using scaling factors, that trade-off compression with accuracy. DoReFa-Net Zhou et al. [2016] further extended the idea of binarization (using the sign function) to gradients as well. They also generalized the method to create networks with arbitrary bit-widths for weights, activations and gradients. ABC-Net Lin et al. [2017] improved upon the ideas from XNOR-Net by using multiple binary weights to approximate the full precision weights (instead of scaling factors) and using multiple binary activations. These changes showed that performance like that of XNOR-Net can be achieved without the scaling factors. Tang et al. [2017] introduced seemingly small but impacting changes to improve accuracy, one of which was the use of a regularization function: $|1 - W^2|$ that carried on to further works. BNN+ Darabi et al. [2018] brought about yet another performance boost by careful regularization strategies and replacing the plain STE with a "SignSwish" activation, a modified version of the Swish-like activation Ramachandran et al. [2017]. Yin et al. [2019] provide key theoretical justification to the use of STE by showing a positive correlation between the true and the estimated "coarse" gradient obtained through STE chain rule.

ADMM algorithm. ADMM is an optimization algorithm that combines the decomposability of dual ascent with the the superior convergence guarantees of the method of multipliers. The algorithm, which is believed to be first introduced by Glowinski and Marroco [1975] and Gabay and Mercier [1976], can be shown to be equivalent to the Douglas-Rachford splitting algorithm Douglas and Rachford [1956]. Boyd et al. [2011] provides a comprehensive overview of the method. Recently, ADMM has sparked the interests of many researchers due to its simplicity, theoretical convergence rates, and parallelization capabilities. The extensibility of ADMM to inexact proximal updates and non-convex problems make it appealing for a lot of problems in machine learning. Hong et al. [2016] is perhaps the first work that extended the analysis of ADMM to nonconvex problems and showed its convergence to first-order stationary points. This analysis is later strengthened in Hong et al. [2018] by showing the convergence of ADMM iterates to second-order stationary points. Another interesting work by Wang et al. [2019b] analyzed the convergence of ADMM for nonconvex and possibly nonsmooth objectives and showed that ADMM, applied to many statistical problems, is guaranteed to converge. In the optimization society, the behavior of ADMM when applied to problems with nonconvex objective functions have also been studied in other regimes such as multiaffine constraints Gao et al. [2020], dynamically changing convex constraints Zhang et al. [2020], finite-sum objective functions, inexact and asynchronous update rules Hong [2017], Zhang and Luo [2018], to name just a few. Wang and Banerjee [2014] generalizes ADMM to Bregman ADMM (BADMM), which allows the choice of different Bregman divergences to exploit the structure of problems. ADMM has also been used as heuristics to solve mixed-integer quadratic programming. Takapoui et al. [2020] proposed an ADMM based algorithm approximately solving convex quadratic functions over the intersection of affine and separable constraints. The Deep Learning community has been no exception to this increased interest in ADMM. Wang et al. [2019a] provided global convergence guarantees for an ADMM-based optimizer for deep neural networks. Ye et al. [2018] used ADMM to devise an effective weight-pruning technique in DNNs for better compression.

With the rising interest in quantization for neural network compression, several works have tried ADMM-based approaches. Leng et al. [2018] were among the first to use an ADMM formulation for weight quantization (not activations) and demonstrated extremely superior results on ImageNet classification. Zhang et al. [2018] proposed a systematic DNN weight pruning framework using ADMM. Ye et al. [2019], Lin et al. [2019], Zhang et al. [2018] extended the work Zhang et al. [2018] by proposing a progressive multi-step approach that not only leads to a better performance, but also can be applied to weight binarization. TP-ADMM Yuan et al. [2019b] used powerful practical improvements to break the training procedure into optimized stages and extend the formulation for binarizing both weights and activations with the state of the art results. Liu et al. [2020] proposed an automatic structured pruning framework, adopting ADMM based algorithm, which boosted the compression ratio to an even higher level. Several works investigated the implementation of ADMM based weight pruning algorithms on hardware level. Yuan et al. [2019a], Ren et al. [2019] explored the idea of algorithm-hardware co-design framework using ADMM. Li et al. [2019] showed ADMM based weight pruning achieved significant storage/memory reduction and speedup in mobile devices with negligible accuracy degradation. In spite of these promising empirical results, the theoretical understanding of ADMM with respect to quantization is still close to non-existent.

2 Problem Formulation

Consider the following discrete optimization problem:

$$\min_{x} \; f(x), \quad \text{s.t.} \quad x \in \mathcal{A} = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d \tag{1}$$

where $\mathcal{A}$ is a discrete subset of $\mathbb{R}^d$. One approach for solving this problem is to sweep across all values in $\mathcal{A}$ and find the optimum point. While this approach results in finding the global optimal solution(s), it is not practical in the quantization procedures of machine learning models. In particular, in this application, the set $\mathcal{A}$ is a discrete grid defined over the space of neural network parameters. Hence, $n = |\mathcal{A}|$ is exponential in the dimension d and it is computationally impossible to sweep over all values of $\mathcal{A}$. While the size of the set $\mathcal{A}$ can be very large, we make an assumption that the projection to the set $\mathcal{A}$ can be done efficiently. To state our assumption clearly, let us formally define the projection operator followed by two clarifying examples.

Definition 2.1. *For any finite set $\mathcal{A}$, the projection of a point x, defined as $\mathcal{P}_{\mathcal{A}}(x)$, is a point $x_p = \arg\min_{a \in \mathcal{A}} \|x - a\|^2$. If the set $\arg\min_{a \in \mathcal{A}} \|x - a\|^2$ is non-singleton, we choose an element in the set with the smallest lexicographical value*[*].

[*] We can break the tie in different ways. We can also pick one of the points in the set $\arg\min_{a \in \mathcal{A}} \|x - a\|^2$ uniformly at random. This choice will make our results to hold with probability one.

Assumption 2.2. *Projection to the set $\mathcal{A}$ can be done in a computationally efficient manner.*

Example 2.3. *Suppose $\mathcal{A} = \{-1, +1\}^d$ in (1) with $|\mathcal{A}| = 2^d$. One can verify that $\mathcal{P}_{\mathcal{A}}(x) = \text{sign}(x) = (\bar{x}_1, \ldots, \bar{x}_d) \in \mathbb{R}^d$ where $\bar{x}_i = +1$ if $x_i \geqslant 0$ and $\bar{x}_i = -1$ if $x_i < 0$. Thus, despite the exponential size of the set $\mathcal{A}$, the projection operator can be computed efficiently.*

Example 2.4. *Assume $\mathcal{A} = \{x \in \mathbb{Z}^d \mid a \leqslant x \leqslant b\}$ with $a, b \in \mathbb{R}^d$ and $\mathbb{Z}$ being the set of integer numbers. Due to the Cartesian product structure of the set $\mathcal{A}$, one can verify that $\mathcal{P}_{\mathcal{A}}(x) = (\bar{x}_1, \ldots, \bar{x}_d)$ with $\bar{x}_i = b_i$ if $x_i > b_i$, $\bar{x}_i = a_i$ if $x_i < a_i$, and $\bar{x}_i = \text{round}(x_i)$ if $a_i \leqslant x_i \leqslant b_i$. Thus, the projection operator can be computed efficiently despite the exponential size of $\mathcal{A}$.*

The above two examples are the constraint sets that appear in the quantization/binarization of machine learning models. Next, we describe the ADMM algorithm for solving optimization problem (1).

3 Alternating Direction Method of Multipliers for Quantization (`ADMM-Q`)

3.1 Review of ADMM

ADMM aims at solving linearly constrained optimization problems of the form

$$\min_{w,z} \; h(w) + g(z) \qquad \text{s.t.} \quad Aw + Bz = c,$$

where $w \in \mathbb{R}^{d_1}, z \in \mathbb{R}^{d_2}$, $c \in \mathbb{R}^k$, $A \in \mathbb{R}^{k \times d_1}$, and $B \in \mathbb{R}^{k \times d_2}$. By forming the augmented Lagrangian function

$$\mathcal{L}(w, z, \lambda) \triangleq h(w) + g(z) + \langle \lambda, Aw + Bz - c \rangle + \frac{\rho}{2} \|Aw + Bz - c\|_2^2,$$

each iteration of ADMM applies alternating minimization to the primal variables and gradient ascent to the dual variables. More precisely, at iteration r, ADMM uses the update rules:

$$\begin{aligned} \text{Primal Update:} \quad & w^{r+1} = \arg\min_{w} \mathcal{L}(w, z^r, \lambda^r), \\ & z^{r+1} = \arg\min_{z} \mathcal{L}(w^{r+1}, z, \lambda^r) \\ \text{Dual Update:} \quad & \lambda^{r+1} = \lambda^r + \rho \left(Aw^{r+1} + Bz^{r+1} - c \right). \end{aligned} \tag{2}$$

As discussed in section 1.1, this algorithm has been well-studied for continuous optimization. Next, we discuss how this algorithm can be used in the discrete optimization problem (1).

3.2 Description of `ADMM-Q`

In order to apply ADMM algorithm to the quantization problem (1), we first re-write (1) as

$$\min_{x} \quad f(x) + \mathcal{I}_{\mathcal{A}}(y) \qquad \text{s.t.} \quad x = y, \tag{3}$$

where $\mathcal{I}_{\mathcal{A}}(y) = 0$ if $y \in \mathcal{A}$, and $\mathcal{I}_{\mathcal{A}}(y) = +\infty$ if $y \notin \mathcal{A}$. Following the steps of regular ADMM in section 3.1, we can update the primal and dual variables alternatingly. The resulting algorithm, which is called Alternating Direction Method of Multipliers for Quantization (`ADMM-Q`), is summarized in Algorithm 1. The details of the derivation of this algorithm can be found in appendix A. Step 4 in this algorithm requires solving an unconstrained optimization problem. In our setting, as we will see later, when ρ is chosen large enough, the function $\mathcal{L}(x, y^{r+1}, \lambda^r)$ is strongly convex in x. Thus solving this problem is assumed to be possible for now. We later relax step 4 to inexact update rule.

Algorithm 1 `ADMM-Q`

1: **Input**: Constant $\rho > 0$; initial points $x^0 = y^0 \in \mathcal{A}$, $\lambda^0 \in \mathbb{R}^d$
2: **for** $r = 0, 1, 2, \ldots$ **do**
3:     **Update** y: $\quad y^{r+1} = \mathcal{P}_{\mathcal{A}}(x^r + \rho^{-1}\lambda^r)$
4:     **Update** x: $\quad x^{r+1} = \arg\min_x \mathcal{L}(x, y^{r+1}, \lambda^r)$
5:     **Update** λ: $\quad \lambda^{r+1} = \lambda^r + \rho(x^{r+1} - y^{r+1})$
6: **end for**

3.3 Convergence Analysis of `ADMM-Q`

In order to analyze the behavior of `ADMM-Q`, we make the following assumptions on f:

Assumption 3.1. *The function f is lower bounded on $\mathcal{A}$. That is, $-\infty < f_{\min} \triangleq \min_{a \in \mathcal{A}} f(a)$.*

Assumption 3.2. *The function f is differentiable and its gradient is L_f–Lipschitz, i.e.,*

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L_f \|x - y\|, \; \forall x, y \in \mathbb{R}^d.$$

Assumption 3.3. *There exist a constant $\mu \geqslant 0$ such that f is μ-weakly convex, i.e. $f(x) + \frac{\mu}{2}\|x\|^2$ is convex.*

When f is twice continuously differentiable, it is easy to verify that $\mu \leqslant L_f$. However, defining these two constants separately will allow us to get tighter bounds for the cases that these two constants are different. Let us also state a few useful lemmas that will help us understand the behavior of `ADMM-Q`. The proofs of these lemmas are relegated to appendix B.

Lemma 3.4. *If $\rho \geqslant L_f$, we have $\mathcal{L}(x^r, y^r, \lambda^r) \geqslant f(y^r) \geqslant f_{\min}, \quad \forall r \geqslant 1$.*

Lemma 3.5. *Define $\sigma(\rho) \triangleq \rho - \mu$. We have*

$$\mathcal{L}(x^{r+1}, y^{r+1}, \lambda^{r+1}) - \mathcal{L}(x^r, y^r, \lambda^r) \leqslant (\rho^{-1}L_f^2 - \frac{\sigma(\rho)}{2}) \left\| x^{r+1} - x^r \right\|^2. \quad (4)$$

This lemma states that by choosing ρ large enough so that $\rho^{-1}L_f^2 - \frac{\sigma(\rho)}{2} < 0$, we ensure the decrease of the augmented Lagrangian function at each iteration[†]. This property combined with Lemma 3.4 implies that $f(y^r) \leqslant \mathcal{L}(x^r, y^r, \lambda^r) \leqslant \mathcal{L}(x^0, y^0, \lambda^0) = f(y^0)$. That is, ADMM-Q cannot output a point worse than the initial point. Next, we use these lemmas to analyze the limiting behavior of the iterates of ADMM-Q. To do that, let us first define the following stationarity concept.

Definition 3.6. *We say a point $\bar{x}$ is a $\rho-$stationary point of the optimization problem* (1) *if*

$$\bar{x} \in \arg\min_{a \in \mathcal{A}} \|a - (\bar{x} - \rho^{-1}\nabla f(\bar{x}))\|.$$

In other words, the point $\bar{x}$ cannot be locally improved using projected gradient descent with step-size ρ^{-1}. Unlike the usual definitions of stationarity for convex constraints, our definition of stationarity depends on the constant ρ. Denoting the set of ρ-stationary solutions with $\mathcal{T}_\rho$, it is easy to see that $\mathcal{T}_{\rho_1} \subseteq \mathcal{T}_{\rho_2}$ when $\rho_1 \leqslant \rho_2$. Thus, in general we would want to have ρ as small as possible. The following lemma justifies the definition of ρ-stationary.

Lemma 3.7. *Assume $x^\star$ is an optimal solution to problem* (1)*, then $x^\star$ is a ρ-stationary point for any $\rho \geqslant L_f$.*

Our ρ-stationarity definition (Definition 3.6) is a natural extension of the continuous setting. It is also closely related to stationarity defined in proximal gradient methods, e.g., see Drusvyatskiy and Lewis [2018], Kadkhodaie et al. [2014], in particular when the proximal operator is associated with an indicator function. Note that, our ρ-stationary definition is a non-trivial necessary condition for optimality. Also when $\rho < L_f$, such stationary points may not exist. See Example B.2 in appendix B.

Remark 3.8. *Because of the fact that Definition 3.6 is a natural extension of the continuous case, it is straightforward to prove the convergence of Projected Gradient Descent (PGD)[‡] algorithm to such stationary set; see appendix D for more details.*

[†]When f is convex, $\mu = 0$ and hence $\sigma(\rho) = \rho$. Thus choosing $\rho > \sqrt{2}L_f$ suffices to ensure the decrease of the augmented Lagrangian function. For the general nonconvex twice differentiable functions, choosing $\rho > 2L_f$ will imply that $\rho^{-1}L_f^2 - \frac{\sigma(\rho)}{2} < 0$, and hence the decrease is guaranteed by Lemma 3.5.

[‡]Each step of PGD comprises of performing a gradient step and then projecting to the feasible set.

Theorem 3.9. *Assume that f satisfies Assumptions 3.1, 3.2 and 3.3. Assume further that ρ is chosen large enough so that $\rho^{-1}L_f^2 - \frac{\sigma(\rho)}{2} < 0$. Let $(\bar{x}, \bar{y}, \bar{\lambda})$ be a limit point of the ADMM-Q algorithm. Then $\bar{x}$ is a ρ–stationary point of the optimization problem* (1).

Remark 3.10. *The previous convergence results for non-convex ADMM, Li and Pong [2015], do not apply to our specific setting. For our problem* (1)*, the stationarity notion defined in equation (4) of Li and Pong [2015] is satisfied for every feasible point since the sub-differential set of every feasible point contains* 0 *(when the feasible set is discrete and finite size). Thus, any feasible point is a stationary point according to the stationary notion in Li and Pong [2015] (see equation (4)). Thus the convergence results and inequalities in [Li and Pong, 2015, Theorem 1] would be vacuous in our setting.*

Remark 3.11. *The convergence results presented in Wang et al. [2019b] do not apply to our setting. This is due to the fact that Wang et al. [2019b] uses Lipschitz sub-minimization paths assumption (Assumption A3). If we specialize their assumption to our setting, their assumption requires that the mappings $H(u) = \arg\min_y f(x) + \mathcal{I}_{\mathcal{A}}(y)$ s.t. $y = u$ and $F(u) = \arg\min_x f(x)$ s.t. $y = u$ are well-defined and Lipschitz continuous. Clearly, Both of these assumptions do not hold in our setting due to non-convexity (and disconnected nature) of the set $\mathcal{A}$. Moreover, regarding global convergence, Theorem 1 and 2 in Wang et al. [2019b] use KL condition (after introducing indicator functions). These assumptions also do not hold in our setting.*

While Theorem 3.9 establishes the convergence of ADMM-Q, this algorithm is far from its inexact version implemented in practice. Next, we analyze the inexact version of ADMM-Q which is used most often in practice and in particular in training binarized neural networks.

4 Inexact ADMM-Q (I-ADMM-Q)

Updating the variable x in ADMM-Q requires finding the minimizer of $\mathcal{L}(\cdot, y^{r+1}, \lambda^r)$; see step 4 in Algorithm 1. Although $\mathcal{L}(\cdot, y^{r+1}, \lambda^r)$ is strongly convex when $\rho > \mu$, finding the exact minimizer might not be practically possible. In practice, we apply iterative methods such as (stochastic) gradient descent to obtain an approximate solution $x^{r+1} \approx \arg\min_x \mathcal{L}(x, y^{r+1}, \lambda^r)$. In this section, we show that ADMM-Q algorithm converges under such an inexact update rule. More precisely, instead of the exact update rule in step 4 of Algorithm 1, we choose a γ–approximate point x^{r+1} that satisfies

$$\|x^{r+1} - x_\star^{r+1}\| \leqslant \gamma \min\left\{\|x^{r+1} - y^{r+1}\|, \|x^{r+1} - x^r\|\right\}, \tag{5}$$

for some positive constant γ. Here $x_\star^{r+1} \triangleq \arg\min_x \mathcal{L}(x, y^{r+1}, \lambda^r)$ is the exact minimizer. The resulting inexact ADMM algorithm, dubbed `I-ADMM-Q`, is summarized in Algorithm 2. Notice that when $\gamma = 0$, this inexact algorithm reduces to the exact `ADMM-Q` algorithm.

Algorithm 2 `I-ADMM-Q`

1: **Input**: Constants $\rho, \gamma > 0$; initial points $x^0 = y^0 \in \mathcal{A}$, $\lambda^0 \in \mathbb{R}^d$
2: **for** $r = 0, 1, 2, \ldots$ **do**
3: **Update** y: $y^{r+1} = \mathcal{P}_{\mathcal{A}}(x^r + \rho^{-1}\lambda^r)$
4: **Update** x by finding a point x^{r+1} satisfying (5)
5: **Update** λ: $\lambda^{r+1} = \lambda^r + \rho(x^{r+1} - y^{r+1})$
6: **end for**

Similar inexactness measures have previously been used in the literature; see, e.g., Li et al. [2018], Reddi et al. [2016]. Notice that since $\mathcal{L}(x, y, \lambda)$ is strongly convex in x, gradient descent algorithm requires only $O(\log(1/\gamma))$ iterations to find a γ-approximate solution. Hence, in practice, we do not need to run many iterations of gradient descent. Next, we present our convergence result for `I-ADMM-Q`.

Theorem 4.1. *Assume that f satisfies Assumptions 3.1, 3.2 and 3.3. Also assume that the iterates of `I-ADMM-Q` are bounded, and the constant ρ and γ are chosen such that*

$$\frac{2L_f^2 + 8(\rho + L_f)^2\gamma^2}{\rho} + \frac{\gamma^2(\rho + L_f) - (1-\gamma)^2\sigma(\rho)}{2} < 0,$$

with $\sigma(\rho) = \rho - \mu$. Then, for any limit point $(\bar{x}, \bar{y}, \bar{\lambda})$ of the iterates, $\bar{x}$ is a ρ–stationary point of (1).

One can verify that the inequality above always holds for $\rho = 6L_f$ and $\gamma \leqslant 0.1$. However, depending on various trade-offs, we may choose different values of γ and ρ.

Remark 4.2. *In practice, checking condition (5) may be impossible since $x_\star^{r+1}$ is not known exactly. To resolve this issue, notice that the strong convexity of $\mathcal{L}(\cdot, y^{r+1}, \lambda^r)$ implies that $\sigma(\rho)\|x - x_\star^{r+1}\| \leqslant \|\nabla_x \mathcal{L}(x, y, \lambda)\|$. Hence, we can use the following checkable sufficient condition instead of (5):*

$$\|\nabla_x \mathcal{L}(x^{r+1}, y^{r+1}, \lambda^r)\| \leqslant \rho\gamma \min\left\{\|x^{r+1} - y^{r+1}\|, \|x^{r+1} - x^r\|\right\}.$$

5 Injecting Randomness to the Algorithm

The analyses in the previous sections only show that the algorithm converges to a stationary solution of the form defined in Definition 3.6. As mentioned earlier, our stationary set includes more points as ρ increase. Thus, to obtain a point satisfying stronger stationary condition, we need to pick the smallest possible ρ. However, reducing the value of ρ beyond certain value results in instability and divergence in `ADMM-Q`, as suggested by our theory and numerical experiments. Another approach that has been utilized in practice to escape spurious stationary solutions is the use of randomness/noise in the algorithm Jin et al. [2017], Lu et al. [2019b], Xu et al. [2018], Allen-Zhu and Li [2018], Barazandeh and Razaviyayn [2018], Lu et al. [2019a]. In order to inject randomness to our algorithm, we propose the following step at each iteration r: draw a set of (potentially correlated) Bernoulli random variables $m^r = \{m_1^r, m_2^r, \ldots, m_d^r\}$. Each m_i^r, corresponds to the coordinate i in vector y with $\text{Prob}(y_i^r = 1) = p_i^r > 0$. Then, we update y_i in iteration r if and only if $m_i^r = 1$. This variant of `ADMM-Q`, which we denote by `ADMM-R`, is presented in Algorithm 3. The convergence result of this algorithm is stated in Theorem 5.1. The proof of this result follows the same steps as in the ones in Theorem 3.9, and hence we omit the proof here.

Algorithm 3 `ADMM-R`

1: **Input**: Constants $\rho, \gamma > 0$; initial points $x^0 = y^0 \in \mathcal{A}$, $\lambda^0 \in \mathbb{R}^d$; the sequence $\{p_i^r\}_{i,r} \geqslant \alpha > 0$.
2: **for** $r = 0, 1, 2, \ldots$ **do**
3: **Generate** m: $m^r = \{m_1^r, m_2^r, \ldots, m_d^r\}$
4: **Compute** $\hat{y}$: $\hat{y}^{r+1} = \mathcal{P}_{\mathcal{A}}(x^r + \rho^{-1}\lambda^r)$
5: **Update** y: $y_i^{r+1} = m_i^r \hat{y}_i^{r+1} + (1 - m_i^r) y_i^r$, $\forall\, i = 1, \ldots, d$
6: **Update** x: $x^{r+1} = \arg\min_x \mathcal{L}(x, y^{r+1}, \lambda^r)$
7: **Update** λ: $\lambda^{r+1} = \lambda^r + \rho(x^{r+1} - y^{r+1})$
8: **end for**

Theorem 5.1. *Assume that the constraint set $\mathcal{A}$ in (1) is a Cartesian product of simple coordinate-wise sets of scalers. Then, under the same set of assumptions as in Theorem 3.9, every iterate of the `ADMM-R` algorithm is a ρ-stationary point of (1).*

Notice that the convergence of this algorithm requires that the set $\mathcal{A}$ to be of the Cartesian product form. This assumption is necessary since the coordinates of y is updated separately; see Powell [1973], Bertsekas [1997], Razaviyayn et al. [2013] for necessity of such an assumption in the presence of coordinate-wise update rule. Having said that, the constraint sets in the quantization context satisfy this assumption as illustrated in Example 2.3 and Example 2.4.

6 ADMM-Q with Soft Projection (ADMM-S)

Step 3 in ADMM-Q algorithm requires projection to the discrete set $\mathcal{A}$. Such a projection is non-continuous which may result in instabilities in the algorithm. As a solution, we can use "soft projection" in ADMM algorithm. To obtain such soft projections, we start by replacing the indicator function $\mathcal{I}_{\mathcal{A}}(\cdot)$ in the objective function with a soft indicator function defined below.

Definition 6.1. *Given a finite set $\mathcal{A} \subseteq \mathbb{R}^d$, we define the Soft Indicator Function $\mathcal{S}_{\mathcal{A}} : \mathbb{R}^d \mapsto \mathbb{R}$ as*

$$\mathcal{S}_{\mathcal{A}}(x) = \min_{a \in \mathcal{A}} \|x - a\|_2.$$

Replacing the indicator function $\mathcal{I}_{\mathcal{A}}(\cdot)$ with the soft indicator function $\mathcal{S}_{\mathcal{A}}$ in (3), we obtain

$$\min_x \quad f(x) + \beta \mathcal{S}_{\mathcal{A}}(y) \quad \text{s.t.} \quad x = y,$$

where $\beta > 0$ is some given constant. Following the steps of ADMM, we obtain the ADMM algorithm with soft projections (ADMM-S), which is summarized in Algorithm 4. The details of the derivation of this algorithm is summarized in appendix E.

Algorithm 4 ADMM-S

1: **Input**: Constant $\rho > 0$, $\beta > 0$; initial points $x^0 = y^0 \in \mathcal{A}$, $\lambda^0 \in \mathbb{R}^d$
2: **for** $r = 0, 1, 2, \ldots$ **do**
3: **Compute**: $z^{r+1} = x^r + \rho^{-1}\lambda^r$, $\tilde{z}^{r+1} = \mathcal{P}_{\mathcal{A}}(z^{r+1})$ and $z_d = \tilde{z}^{r+1} - z^{r+1}$
4: **Update** y:
5: $$y^{r+1} = \begin{cases} z^{r+1} + \dfrac{\rho^{-1}\beta z_d}{\|z_d\|_2} & , \ \rho^{-1}\beta \leq \|z_d\|_2 \\ \tilde{z}^{r+1} & , \ \rho^{-1}\beta > \|z_d\|_2 \end{cases}$$
6: **Update** x: $x^{r+1} = \arg\min_x \mathcal{L}(x, y^{r+1}, \lambda^r)$
7: **Update** λ: $\lambda^{r+1} = \lambda^r + \rho(x^{r+1} - y^{r+1})$
8: **end for**

As shown in appendix E, this algorithm coincides with ADMM-Q if β is chosen large enough. However, for small values of ρ, this algorithm results in a different trajectory. In this case, while the iterates of the algorithm does not necessarily converge to the set $\mathcal{A}$, the y iterates are kept close to set $\mathcal{A}$. Moreover, as shown in appendix E, the augmented Lagrangian function is monotonically decreasing and it converges. Finally, we would like to mention that similar hard and soft indicators have been used before for sparse signal recovery through soft and hard thresholding operators Donoho [1995], Blumensath and Davies [2008].

7 Numerical Experiments

We empirically evaluate the performance of the proposed algorithms in the following two problems: 1) Solving quadratic optimization problems with integer constraints. 2) Training quantized neural networks. The link to code is available in appendix H.

7.1 Numerical Experiment on Quadratic Optimization with Integer Constraints

In this experiment, we use the presented algorithms (ADMM-Q and its variants) to solve the optimization problem

$$\min_x \quad \frac{1}{2} x^\top Q x + b^\top x \quad \text{s.t.} \ \ x \in \mathcal{A} \triangleq v\mathbb{Z}^d, \tag{6}$$

for some given $Q \in \mathbb{R}^{d \times d}$, $b \in \mathbb{R}^d$, and $v \in \mathbb{Z}^+$. Here, the constraint set enforces that the solution should be an integer number which is a multiple of v. We generate matrix Q via the equation $Q = \widetilde{Q}^\top \widetilde{Q} + \widetilde{q}\widetilde{q}^\top$, where $\widetilde{Q}_{ij} \sim N(0,1)$, $\widetilde{q}_i \sim N(0, \sigma_{\widetilde{q}}^2)$, $1 \leq i, j \leq d$. Note that the Lipschitz constant of the objective function (parameter L_f in the previous sections) can be adjusted through changing $\sigma_{\widetilde{q}}^2$. We compare the performance of projected gradient gradient descent (PGD), GD+Proj, ADMM-Q, ADMM-S, and ADMM-R for different values of d and $\sigma_{\widetilde{q}}^2$ (see appendix F for more details). The PGD algorithm is defined through the iterative update rule $x^{r+1} = \mathcal{P}_{\mathcal{A}}(x^r - \rho^{-1}\nabla f(x^r))$. The "GD+Proj" algorithm, runs gradient descent to find the global optimum of unconstrained problem, then it projects the final solution onto the feasible set $\mathcal{A}$.

For each problem instance, we run each algorithm initialized at the same random point for 30,000 iterations (except 100,000 iterations for PGD to make sure it is convergent). The best objective value over the last 50 iterations of the algorithm will be recorded as the result of each run. We repeated this procedure for 50 different initilizations, and compute the median, 25% quartile and 75% quartile over 50 runs. We use the best hyper-parameter for each algorithm by median, and report the median, 25% quartile and 75% quartile. The list of hyper parameters used can be found in appendix F.

Results. We only report our results for $(v, d, \sigma_{\widetilde{q}}^2) = (8, 16, 30)$ here. More simulations can be found in appendix F. Figure 1 shows the performance of the studied algorithms for five different problem instances. Each point on x-axis represents one problem instance; and y-axis is the final obtained objective value. As expected, ADMM-Q outperforms PGD and GD+Proj with large margins. We also observe that both ADMM-S and ADMM-R have better median final objective values than ADMM-Q. In addition, the final objective value has a

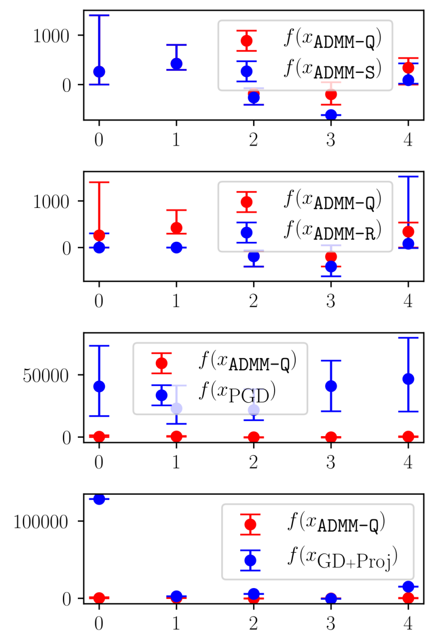

Figure 1: Performance of ADMM-Q, ADMM-S, ADMM-R and PGD on different problem instances

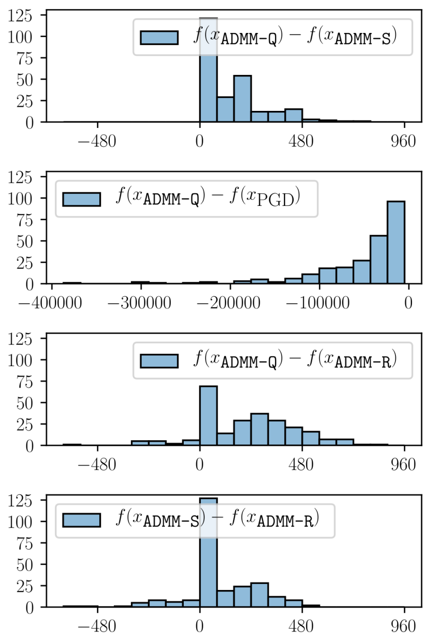

Figure 2: Histogram of the difference of obtained objective values for different algorithm pairs

smaller variance in these two algorithms. More importantly, the median tends to overlap with the 25% quantile, i.e., the objective of at least 25 runs are almost the same as the minimum objective over 50 runs.

To better understand the performance gap between different algorithm for the same initialization, we conducted one additional experiment: we generated 5 instance of Q and b, and for each instance we ran 50 different random initialization, resulting in 250 total runs. We recorded the final objective value by each algorithm. Then we computed the differences between the objective values obtained by two algorithms for the same initialization. We plot the histograms of these differences in Figure 2. In this plot, $f(x_{\texttt{ADMM-Q}})$ denotes the final objective value obtained by `ADMM-Q` algorithm (similar notation is used for other algorithms). Our histogram plot suggests that `ADMM-S` and `ADMM-R` outperform `ADMM-Q` for almost all 250 runs. It also shows that PGD performs much worse than `ADMM-Q` or its variants. We also observe that `ADMM-R` slightly outperforms `ADMM-S`.

7.2 Neural Network Binarization

While ADMM algorithm has been extremely successful in binarization and pruning of neural networks Ye et al. [2018, 2019], Yuan et al. [2019b,a], Leng et al. [2018], Lin et al. [2019], Zhang et al. [2018], Liu et al. [2020], Ren et al. [2019], Li et al. [2019], most of these works combine ADMM with other heuristics. To understand the behavior of the ADMM algorithm (independent of other heuristics), here we study the performance of pure `ADMM-Q` and its variants (with no additional heuristics) when used for binarizing neural networks trained on MNIST and CIFAR-10 datasets.

7.2.1 MNIST

The MNIST dataset LeCun et al. [1998] consists of 28×28 arrays of grayscale pixel images classified into 10 handwritten digits. It includes $60,000$ training images and $10,000$ testing images. The task here is to train a binary-weighted classifier to recognized hand written digits, which can be formulated as

$$\min_{W} \frac{1}{N} \sum_{i=1}^{N} \ell(f(W; x_i), y_i) \text{ s.t. } W \in \{-1, +1\}^d \quad (7)$$

where (x_i, y_i) is the i-th training sample; x_i is the input image; y_i is the label; W represents the weights of the network. The work in Courbariaux et al. [2015] used "Straight Through Estimator" to binarize the network and reached the accuracy level of the full-precision network. We repeat the experiment with the same network as Courbariaux et al. [2015] and apply `ADMM-Q` and its variants. Similar to the quadratic case, we also compare the performance with PGD and GD+Proj. We conduct two sets of experiments: with pretraining and without pretraining. To the best of our

Algorithm	Accuracy
BinaryConnect[§]	98.71%
Full Precision	$98.87 \pm 0.04\%$
GD+Proj	$74.92 \pm 4.83\%$
PGD	$92.73 \pm 0.23\%$
ADMM-Q	$98.21 \pm 0.16\%$
ADMM-R	$97.78 \pm 0.23\%$
ADMM-S	$98.21 \pm 0.07\%$

Table 1: Testing accuracies for MNIST dataset

Algorithm	Accuracy
Progressive DNN[¶]	93.53%
Full Precision	93.06%
GD+Proj	9.86%
PGD	63.53%
ADMM-Q	82.74%
ADMM-R	84.87%
ADMM-S	84.72%

Table 2: Testing accuracies for CIFAR-10 dataset

	Full-precision	Binary
MNIST	140.55 MB	4.53 MB
CIFAR-10	53.53 MB	1.72 MB

Table 3: The storage savings of binarized neural networks

knowledge, all the ADMM-based approaches Ye et al. [2018, 2019], Yuan et al. [2019b,a], Leng et al. [2018], Lin et al. [2019], Zhang et al. [2018], Liu et al. [2020], Ren et al. [2019], Li et al. [2019] start from a pre-trained full-precision network. However, in order to solely study the performance of ADMM-based methods (and not additional modules around it), we avoid using pre-training in some of our experiments. We also did not use any popular heuristics and we relies on implementing our plain ADMM-based algorithms.

To remove the effect of random initialization, we run each algorithm for 5 times and record the mean and standard deviation of the testing accuracy. For algorithms with pre-training, we pre-train the model with full precision and then apply the algorithm. Training parameters and network structures be found in Table 9 and Table 8 in the appendix. Adam optimizer is used for all algorithms. Note that in step 3 of algorithm 1, it is required to solve a minimization problem, which is not always tractable in practice. Thus, here we apply 5 epochs of Adam update on W.

Results. Table 1 shows that plain `ADMM-Q` and its variants have comparable results with BinaryConnect Courbariaux et al. [2015]. Binarizing the weights saves the storage as much as 96.78% (See Table 3). One substantial difference between Courbariaux et al. [2015] and the proposed work is that we do not use any heuristics and the proposed algorithm enjoys theoretical guarantees. Note that for `ADMM-Q` without pre-training, we fix a value of ρ and keep it until the end of the training process. We observed that one can indeed use "scheduling" for parameter ρ, i.e., increasing it gradually, to shorten the training time. It is worth mentioning that pre-training (with non-binarized weights) in fact further improves the performance of ADMM-based methods (see appendix G).

7.2.2 CIFAR-10

The CIFAR-10 dateset Krizhevsky et al. [2009] is a collection of images widely used to train machine learning models. It consists of 32×32 sized RGB images classified into 10 mutually exclusive categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset consists of 50,000 training images and 10,000 testing images. Similar to the MNIST experiments, the task is to build a binary-weighted network for classifying the images.

We repeat run our algorithms to train neural networks on CIFAR-10 dataset with and without pretraing. For this experiment, we use Resnet-18 He et al. [2016] architecture. The hyper-parameters used in our experiments are summarized in Table 10. Adam optimizer is used for all algorithms; We apply 25 epochs of Adam updates on W to solve the minimization problem in step 3 of algorithm 1.

Results. Table 2 shows the results of the experiments on CIFAR-10. Binarizing the weights saves the storage of up to 96.79% (See Table 3). Progressive DNN Ye et al. [2019] is the state-of-the-art result which has multiple re-training heuristics involved. One thing worth mentioning here is that we do not use any heuristics. We only use diminishing step-size and increasing rho during the training procedure which is standard. We can see that without pretraining, the results of ADMM-type algorithm are much better than PGD which is consistent with the observations in MNIST and quadratic experiments. `ADMM-R` and `ADMM-S` slightly outperform `ADMM-Q`. Pretraining can further improve the performances of ADMM-type algorithms, making them comparable with the full-precision network (see appendix G).

[§] BinaryConnect Courbariaux et al. [2015]

[¶] Progressive DNN Ye et al. [2019]

Acknowledgments

The authors would like to thank Mingyi Hong (University of Minnesota) for the fruitful discussions and invaluable comments that improved the quality of this paper.

References

Z. Allen-Zhu and Y. Li. Neon2: Finding local minima via first-order oracles. In *Advances in Neural Information Processing Systems*, pages 3716–3726, 2018.

B. Barazandeh and M. Razaviyayn. On the behavior of the expectation-maximization algorithm for mixture models. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 61–65. IEEE, 2018.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

T. Blumensath and M. E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008.

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

M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia. Bnn+: Improved binary network training. *arXiv preprint arXiv:1812.11800*, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

D. L. Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.

W. Gao, D. Goldfarb, and F. E. Curtis. Admm for multiaffine constrained optimization. *Optimization Methods and Software*, 35(2):257–303, 2020.

R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

M. Hong. A distributed, asynchronous, and incremental algorithm for nonconvex optimization: an admm approach. *IEEE Transactions on Control of Network Systems*, 5(3):935–945, 2017.

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

M. Hong, M. Razaviyayn, and J. Lee. Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks. In *International Conference on Machine Learning*, pages 2009–2018, 2018.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1724–1732. JMLR. org, 2017.

M. Kadkhodaie, M. Sanjabi, and Z.-Q. Luo. On the linear convergence of the approximate proximal splitting method for non-smooth convex optimization. *Journal of the Operations Research Society of China*, 2(2):123–141, 2014.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

G. Li and T. K. Pong. Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization*, 25(4):2434–2460, 2015.

H. Li, N. Liu, X. Ma, S. Lin, S. Ye, T. Zhang, X. Lin, W. Xu, and Y. Wang. Admm-based weight pruning for real-time deep learning acceleration on mobile devices. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pages 501–506, 2019.

T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

S. Lin, X. Ma, S. Ye, G. Yuan, K. Ma, and Y. Wang. Toward extremely low bit and lossless accuracy in dnns with progressive admm. *arXiv preprint arXiv:1905.00789*, 2019.

X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *AAAI*, pages 4876–4883, 2020.

S. Lu, M. Hong, and Z. Wang. Pa-gd: On the convergence of perturbed alternating gradient descent to second-order stationary points for structured nonconvex optimization. In *International Conference on Machine Learning*, pages 4134–4143, 2019a.

S. Lu, M. Razaviyayn, B. Yang, K. Huang, and M. Hong. Snap: Finding approximate second-order stationary solutions efficiently for non-convex linearly constrained problems. *arXiv preprint arXiv:1907.04450*, 2019b.

M. J. Powell. On search directions for minimization algorithms. *Mathematical programming*, 4(1):193–201, 1973.

P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

S. J. Reddi, J. Konečný, P. Richtárik, B. Póczós, and A. Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 925–938, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2013.

R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. *International Journal of Control*, 93(1):2–12, 2020.

W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI conference on artificial intelligence*, 2017.

H. Wang and A. Banerjee. Bregman alternating direction method of multipliers. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2816–2824. Curran Associates, Inc., 2014.

J. Wang, F. Yu, X. Chen, and L. Zhao. Admm for efficient deep learning with global convergence. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 111–119, 2019a.

Y. Wang, W. Yin, and J. Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019b.

Y. Xu, R. Jin, and T. Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *Advances in Neural Information Processing Systems*, pages 5530–5540, 2018.

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

S. Ye, X. Feng, T. Zhang, X. Ma, S. Lin, Z. Li, K. Xu, W. Wen, S. Liu, J. Tang, et al. Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm. *arXiv preprint arXiv:1903.09769*, 2019.

P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

G. Yuan, X. Ma, C. Ding, S. Lin, T. Zhang, Z. S. Jalali, Y. Zhao, L. Jiang, S. Soundarajan, and Y. Wang. An ultra-efficient memristor-based dnn framework with structured weight pruning and quantization using admm. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019a.

Y. Yuan, C. Chen, X. Hu, and S. Peng. Tp-admm: An efficient two-stage framework for training binary neural networks. In *International Conference on Neural Information Processing*, pages 580–588. Springer, 2019b.

J. Zhang and Z.-Q. Luo. A proximal alternating direction method of multiplier for linearly constrained nonconvex minimization. *arXiv preprint arXiv:1812.10229*, 2018.

T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

Y. Zhang, E. Dall'Anese, and M. Hong. Online proximal-admm for time-varying constrained convex optimization. *arXiv preprint arXiv:2005.03267*, 2020.

S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.