Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM

Cong Leng, Hao Li, Shenghuo Zhu, Rong Jin
Alibaba Group, Hang Zhou, China
{lengcong.lc, lihao.lh, shenghuo.zhu, jinrong.jr}@alibaba-inc.com

Abstract

Although deep learning models are highly effective for various tasks, such as detection and classification, the high computational cost prohibits the deployment in scenarios where either memory or computational resources are limited. In this paper, we focus on model compression and acceleration of deep models. We model a low bit quantized neural network as a constrained optimization problem. Then, with the use of ADMM, we decouple the discrete constraint and parameters of network. We also show how the resulting subproblems can be efficiently solved with extragradient and iterative quantization. The effectiveness of the proposed method has been demonstrated in extensive experiments on convolutional neural network for image recognition, object detection, and recurrent neural network for language model.

1 Introduction

These years have witnessed the success of convolutional neural networks (CNNs) in a wide range computer vision tasks, such as image classification, object detection and segmentation. Beyond CNNs, recurrent neural networks (RNNs) based models, such as LSTM, also make good progress in nature language processing and speech recognition. The success of deep learning largely owes to the fast development of computing resources. For example, most of the deep learning models are trained on high-ended GPUs or CPU clusters. On the other hand, deeper networks typically impose heavy storage footprint due to the enormous amount of network parameters. For example, the 8-layers Alexnet involves 250 MBytes of model parameters while the 16-layers VGG involves 528 MBytes of model parameters. Both the high computational and storage cost become impediments to popularize the deep neural networks to scenarios where either memory or computational resources are limited. The great interest to deploy deep learning systems on low-ended devices motives the research in compressing deep models to have smaller computation cost and memory footprints.

Considerable efforts have been mounted to reduce the model size and speed up the inference of deep models. Denil et al. [7] pointed out that network weights have a significant redundancy, and tried to reduce the parameters by exploiting the linear structure of network weights. Motivated by this observation, a series of works, e.g. [37, 8, 22, 19], have been proposed based on low-rank matrix/tensor factorization. Gong et al. [12] proposed to quantized the weights of fully connected layers with structured vector quantization techniques. Extending this idea to convolutional layers, Wu et al. [35] obtained a clear gain over the matrix/tensor factorization methods. Han et al. [13] presented the deep compression method which integrates the sparse pruning, vector quantization and Huffman coding, and reduce the model storage by 35x-50x. Other seminal works attempt to restrict the weights to be low precision. For example, the pretrained weights are quantized into a few bits, usually 4-12 bits, resulting in a minimal loss of performance [25, 10, 4, 24]. To the extreme, training low-bits networks from scratch with binary (+1 and -1) or ternary (+1, 0 and -1) weights is attempted in [5, 6, 28, 16, 3].

In this work, we present a unified learning strategy for extremely low-bits neural networks. Formally, we restrict the weights of network to be either zero of powers of two. The advantage of restricting the weights to zero or powers of two is that an expensive floating-point multiplication operation can then be replaced by a sequence of cheaper and faster binary bit shift operations of fixed-point numbers. It is easy to find that training such kind of low-bits neural network can be modeled as a discretely constrained nonconvex optimization problem, which in particular is well known as mixed integer program (MIP). MIPs are NP-hard problems in general with much interest in finding bounds or approximate solutions for them. Inspired from the efficient heuristics proposed to solve MIPs [33, 1, 36], in this paper, we proposed to learn low-bits neural network in the framework of alternating direction method of multipliers (ADMM) [15, 2].

The main idea behind our method is to decouple the continuous variables and discrete constraints with ADMM. Although ADMM has been originally developed for convex problems, there has been much recent interest in applying it to the nonconvex nonsmooth problems, with the analysis of convergence available in some cases [34]. It has been suggested in [33] that, by decoupling the discrete constraints in MIP, one can use the information from dual problem through ADMM to get a better upper bound. In addition, with the proposed framework the difficult objective of low-bits neural network can be decomposed into several subproblems, which are much easier to be solved as we will show. The main contributions of this paper can be summarized as follows:

- We model the low-bits neural network as a discretely constrained nonconvex optimization problem, and introduce auxiliary variables to decouple the discrete constraint based on ADMM algorithm. The originally hard problem are decomposed into several subproblems.

- We show how the resulting subproblems can be efficiently solved with novel extragradient and iterative quantization algorithm. The proposed algorithms enjoy a fast convergence in practice.

- We apply the proposed method to both convolutional neural networks and recurrent neural networks. Extensive experiments on various tasks including image classification, object detection and language model demonstrate the proposed method can advance the state-of-the-art.

2 Backgrounds

In this section, we have a brief review of the representative techniques for compressing and accelerating deep neural network. We also briefly introduce the ADMM algorithm and its nonconvex extension.

2.1 Acceleration and compression of DNN

Acceleration and compression of deep neural networks have attracted much attention these years. Existing methods can be roughly categorized into three schemes, i.e., designing more compact architecture, low-rank matrix (tensor) decomposition, and quantization of weights.

Designing more compact architecture: In the old fashion networks, such as Alexnet [21] and VGG16/19 [30], fully-connected layers involve massive network parameters. In order to get rid of this part of storage, more recent GoogLenet [32] and ResNet [14] propose to replace the fully connected layer with global average pooling, and find this simplification does not sacrifice any accuracy on image classification and object detection tasks. In SqueezeNet [17], the authors propose to replace majority of the 3×3 filters with 1×1 filters since the later has 9x fewer parameters than the former. SqueezeNet achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. He et al. [14] propose the bottleneck structure by replacing two layers of 3×3 convolutions with three layers 1×1, 3×3 and 1×1 convolutions, in which the 1×1 layers are responsible for reducing and then increasing dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. Similar structures are also widely used in Inception-v4 [31] and Darknet [29].

Low-rank matrix (tensor) decomposition: Denton et al. [8] for the first time propose to exploit the linear structure of convolutional filters. This work compresses each convolutional layer by finding an low-rank approximation for the 4 dimensional convolutional weight tensors, and then finetune upper layers with all those below them fixed until the original accuracy is restored. Lebedev et al. [22] propose to utilize CP-decomposition on 4 dimensional convolutional kernels to speed-up character-classification CNN. Tucker decomposition [19] is also used to speed up the execution of CNN models. Instead of approximating linear filters or linear responses, Zhang et al. propose to take the nonlinear units into account. With nonlinear asymmetric reconstruction, the accumulated error of multiple layers can be efficiently reduced [37]. These works showed high speed-up and energy reduction on smartphone equipped with a mobile GPU.

Quantization of connections: Quantized inference using reduced precision arithmetic has been well studied these days. Gong et al. [12] first find that the weights of fully connected layers can efficiently quantized with product quantization. Wu et al. [35] extend the product quantization to both both filter kernels in convolutional layers and weighting matrices in fully-connected layers, and introduce an efficient test-phase computation process with the network parameters quantized. Low-bits quantization based methods are also investigated by several recent works. These works are the most related to our proposed approach. BinaryConnect [5] is the most representative work which use binary weights for forward computation while use real-valued version of weight to accumulate the gradient during backward. BinaryNets [6] is an extension of BinaryConnect. In BinaryNets, not only the weights but also activations of the network are binarized. This can largely accelerate the inference of networks since the expensive 32-bit floating point multiply operation can be replaced by very cheap 1-bit xnor operation. However, these methods achieve good results only on small datasets such as MNIST, CIFAR10 and SVHN. Applying these methods to large scale datasets, such as ImageNet, results in very poor accuracy. For this performance issue, many efforts have been done. For example, XNOR-net [28] proposes to improve the performance of BNN with a better approximation by introducing scale factors for both weights and activations during binarization. DoReFa-net [38] aims at boosting the performance by approximating the activations with more bits.

2.2 ADMM and its non-convex extension

Alternating Direction Method of Multipliers (ADMM) is an algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers. The algorithm solves problems in the form:

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c \end{aligned} \tag{1}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$ and $c \in \mathbb{R}^p$.

The augmented Lagrangian of Eq.(1) can be formed as:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2 \tag{2}$$

where y is the Lagrangian multipliers, and ADMM consists of three step iterations:

$$\begin{aligned} x^{k+1} :=& \quad \arg\min_x L_\rho(x, z^k, y^k) \\ z^{k+1} :=& \quad \arg\min_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} :=& \quad y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Even though ADMM was originally introduced as a tool for convex optimization problems, it turns out to be a powerful heuristic method even for NP-hard nonconvex problems

[2]. Recently, this tool has been used as a heuristic to find approximate solutions to nonconvex problems [33, 1]. In [9], the authors study the Divide and Concur algorithm as a special case of a message-passing version of the ADMM, and introduce a three weight version of this algorithm which greatly improves the performance for some nonconvex problems such as circle packing and the Sudoku puzzle.

3 The Proposed Method

Let us first define the notion in this paper. Denote $f(W)$ as the loss function of a normal neural network, where $W = \{W_1, W_2, \cdots, W_L\}$. W_i denotes the weights of the i-th layer in the network, which for example can be a 4-dimension tensor in convolutional layer or a 2-dimension matrix in fully connected layer. For the simplicity of notation, we regard all the entries in W_i as a d_i-dimension vector in $\mathbb{R}^{d_i}$, and take W as the concatenation of these vectors so that $W \in \mathbb{R}^d$ with $d = \sum_i d_i$.

3.1 Objective function

In this work, we training low-bits quantized neural networks. Specially, we restrict the weights of the network to be either zero or powers of two so that the expensive floating-point multiplication operation can be replaced by cheaper and faster bit shift operation. We realize such a problem can be modeled as constrained optimization, or in particular, mixed integer program. For example, the weights in a *ternary* neural network are restricted to $-1, 0$ or $+1$. In this perspective, training a ternary neural network can be mathematically formulated as constrained optimization:

$$\min_W \quad f(W) \qquad \text{s.t.} \quad W \in \mathcal{C} = \{-1, 0, +1\}^d$$

More generally, the constraints have a form of

$$\mathcal{C} = \{-2^N, \cdots, -2^1, -2^0, 0, +2^0, +2^1, \cdots, +2^N\}$$

where N is an integer which determines the number of bits. Since batch normalization [18] is often used to improve the convergence of deep neural network training, it will lead to a scale invariance in the weight parameters. As a result, we further introduce a scaling factor α to the constraints, i.e., instead of requiring $\mathcal{C} = \{\cdots, -2, -1, 0, +1, +2. \cdots\}$, we simply restrict $\mathcal{C}$ to $\mathcal{C} = \{\cdots, -2\alpha, -\alpha, 0, +\alpha, +2\alpha, \cdots\}$ with an arbitrary scaling factor $\alpha > 0$ that is strictly positive. It is worthy noting that the scale factor α in various layers can be different. That is to say, for a neural network with L layers, we actually introduce L different scaling factors $\{\alpha_1, \alpha_2, \cdots, \alpha_L\}$. Formally, the objective function of the proposed low-bits quantized neural networks can be formulated as:

$$\begin{aligned} \min_W \quad & f(W) \\ \text{s.t.} \quad & W \in \mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_L \end{aligned} \tag{3}$$

where $\mathcal{C}_i = \{0, \pm\alpha_i, \pm 2\alpha_i, \cdots, \pm 2^N \alpha_i\}$ and $\alpha_i > 0$.

3.2 Decouple with ADMM

The optimization in Eq.(3) is non-convex and NP-hard in general because of the existence of discrete constraints. Many heuristics have been proposed to deliver a good but suboptimal solution. For example, BinaryNets [5, 6] replace the weights W with $\mathrm{sign}(W)$ in the forward process, and replace the sign function with hard tanh in the backpropagation step to overcome the problem that gradients of sign function are zero almost everywhere.

In this paper, we discuss a method in which we introduce an auxiliary variable so that the parameters in the objective and the discrete constraints can be decoupled. This is used with ADMM, which will result in the effect that the discrete variables being decoupled when we consider their minimization. The basic idea is largely motivated by the successful application of ADMM in mixed integer program [33, 1, 36, 11].

First of all, the objective in Eq.(3) can be written as

$$\min_{W} \quad f(W) + I_{\mathcal{C}}(W) \tag{4}$$

where $I_{\mathcal{C}}$ is defined as an indicator function for whether $W \in \mathcal{C}$, i.e., if $W \in \mathcal{C}$ then $I_{\mathcal{C}}(W) = 0$, otherwise $I_{\mathcal{C}}(W) = +\infty$. By introducing an auxiliary variable G, we can rewrite the optimization problem in Eq.(4) with an extra constraint such that instead of having the discrete variable in the objective, we will have it in the constraints:

$$\begin{aligned} \min_{W,G} \quad & f(W) + I_{\mathcal{C}}(G) \\ \text{s.t.} \quad & W = G \end{aligned} \tag{5}$$

We now consider the non-convex optimization problem with convex linear constraints. The augmented Lagrange of Eq.(5), for parameter $\rho > 0$, can be formulated as:

$$L_\rho(W, G, \mu) = f(W) + I_{\mathcal{C}}(G) + \frac{\rho}{2}\|W - G\|^2 + \langle \mu, W - G \rangle \tag{6}$$

where μ denotes the Lagrangian multipliers and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. With some basic collection of terms and a change of variable $\lambda = (1/\rho)\mu$, Eq.(6) can be equivalently formed as:

$$L_\rho(W, G, \lambda) = f(W) + I_{\mathcal{C}}(G) + \frac{\rho}{2}\|W - G + \lambda\|^2 - \frac{\rho}{2}\|\lambda\|^2 \tag{7}$$

Following the standard process of ADMM, this problem can be solved by repeating the following iterations:

$$W^{k+1} := \quad \arg\min_{W} \; L_\rho(W, G^k, \lambda^k) \tag{8}$$

$$G^{k+1} := \quad \arg\min_{G} \; L_\rho(W^{k+1}, G, \lambda^k) \tag{9}$$

$$\lambda^{k+1} := \quad \lambda^k + W^{k+1} - G^{k+1} \tag{10}$$

which is respectively the proximal step, projection step and dual update.

3.3 Algorithm subroutines

In this section, we elaborate on how the consequent subproblems in the above algorithm can be efficiently solved.

3.3.1 Proximal step

For the proximal step, we need to find the weights that minimize

$$L_\rho(W, G^k, \lambda^k) = f(W) + \frac{\rho}{2}\|W - G^k + \lambda^k\|^2 \tag{11}$$

Benefit from the decouple of ADMM, we are dealing with an unconstrained objective here. The loss can be interpreted as a normal neural network with a special regularization. Naturally, this problem can be solved with standard gradient decent method. It is easy to obtain the gradient with respect to the weights W:

$$\partial_W L = \partial_W f + \rho(W - G^k + \lambda^k) \tag{12}$$

However, we find the vanilla gradient descent method converges slowly in this problem. Since the second quadratic term occupies a large proportion of the whole lost, SGD will quickly pull the optimizer to the currently quantized weights so that the second term vanishes, and stack in that point. This results in a suboptimal solution as the loss of neural network is not sufficiently optimized.

In order to overcome this challenge, we resort to the extragradient method [20, 27]. An iteration of the extragradient method consists of two very simple steps, prediction and correction:

$$\begin{aligned} W^{(p)} &:= W - \beta \partial_W L(W), \\ W^{(c)} &:= W - \beta \partial_W L(W^{(p)}) \end{aligned}$$

where β is the learning rate. A distinguished feature of the extragradient method is the use of an additional gradient step which can be seen as a guide during the optimization process. Particularly, this additional iteration allows to foresee the geometry of the problem and take the curvature information into account. Specific to our problem, here we provide a more intuitive understanding of the above iterations. For the prediction step, the algorithm will quickly move to a point close to $G^k - \lambda^k$ so that the loss of quadratic regularization vanishes. Then in the correction step, the algorithm moves another step which tries to minimize the loss of neural network $f(W)$. These two steps avoid the algorithm stacking into a less valuable local minima. In practice, we find this extragradient method largely accelerate the convergence of the algorithm.

3.3.2 Projection step

For the projection step, by neglecting those terms that do not depend on G, the loss function can be rewritten as

$$\begin{aligned} \min_{G} \quad & \|G - W^{k+1} - \lambda^{k}\|^2 \\ \text{s.t.} \quad & G \in \mathcal{C} \end{aligned} \tag{13}$$

The key observation of (13) is that while minimizing over G, all the components G_i are decoupled, therefore the auxiliary variables of each layer can be optimized independently. Recall that $W_i, G_i, \lambda_i, \mathcal{C}_i$ denote the weights, auxiliary variables, Lagrangian multipliers and constraints of the i-th layer respectively. We are essentially looking for the projection of $(W_i^{k+1} + \lambda_i^k)$ onto a discrete set $\mathcal{C}_i$. Due to the scaling factor α_i and the constraint being discrete and non-convex, this optimization is nontrivial.

For convenience, we denote $(W_i^{k+1} + \lambda_i^k)$ as V_i. The projection of V_i onto $\mathcal{C}_i$ can be formulated as

$$\begin{aligned} \min_{G_i, \alpha_i} \quad & \|V_i - G_i\|^2 \\ \text{s.t.} \quad & G_i \in \{0, \pm\alpha_i, \pm 2\alpha_i, \cdots, \pm 2^N \alpha_i\}^{d_i}, \quad \alpha_i > 0 \end{aligned} \tag{14}$$

It is easy to find that the objective can be rewritten as:

$$\begin{aligned} \min_{Q, \alpha_i} \quad & \|V_i - \alpha_i \cdot Q\|^2 \\ \text{s.t.} \quad & Q \in \{0, \pm 1, \pm 2, \cdots, \pm 2^N\}^{d_i}, \quad \alpha_i > 0 \end{aligned} \tag{15}$$

We propose an iterative quantization method to get the solution, i.e., optimizing α_i with Q fixed and vice verse. In specific, with Q fixed, the optimal α_i can be easily obtained as

$$\alpha_i = \frac{V_i^T Q}{Q^T Q} \tag{16}$$

With α_i fixed, the optimal Q is

$$Q = \Pi_{\{0, \pm 1, \pm 2, \cdots, \pm 2^N\}} \left(\frac{V_i}{\alpha_i} \right) \tag{17}$$

where Π denotes the projection of $\frac{V_i}{\alpha_i}$ onto $\{0, \pm 1, \pm 2, \cdots, \pm 2^N\}$. The projection onto a discrete set is simply finding the closest point in it. We find such a simple algorithm converges very fast. In most cases, we only need less than five iterations to get a stable solution.

3.3.3 Dual update

The iterate λ^{k+1} can be interpreted as a scaled dual variable, or as the running sum of the error values $W^{k+1} - G^{k+1}$.

4 Experiments

In this section, we evaluate the proposed method on two benchmarks: ImageNet for image classification and Pascal VOC for object detection.

4.1 Image Classification

4.1.1 Implementation Details

In the ImageNet (ILSVRC2012) experiments, all the images are resized to 256×256. The images are then randomly clipped to 224×224 patches with mean subtraction and randomly flipping. No other data augmentation tricks are used in the learning process.

We comprehensively evaluate our method on three models, i.e., Alexnet [21], ResNet-18 [14] and ResNet-50 [14]. For Alexnet, we add batch normalization [18] before each layer to accelerate the convergence of network. For Alexnet and ResNet-18, the batchsizes are set to be 256. For ResNet-50, the batchsize is set to be 128. For all tested networks, the weight decay is 0.0001 and learning rate starts at 0.01. We report both the top-1 and top-5 classification accurate rates on the validation set, using single-view testing (central patch only).

We studied different kinds of bit-width for weight quantization. In specific, we tried binary quantization, ternary quantization, one-bit shift quantization and two-bits shift quantization. For one-bit shift quantization, the weights are restricted to be {-2a, -a, 0, +a, +2a}, which we denoted as {-2, +2} in the comparison. Similarly, two-bits shift quantization are denoted as {-4, +4}. For binary and ternary quantization, we compare the proposed method to state of the arts methods Binary Weight Network (BWN) [28] and Ternary Neural Network (TWN) [23], respectfully.

4.1.2 Results and Analysis

Results on Alexnet: The results on Alexnet are shown in Table 1. AlexNet has 5 convolutional layers and 3 fully-connected layers. The results of BWN and TWN are copied from their corresponding paper. From the comparison between our binary quantization and BWN, we find the the proposed method marginally outperform BWN in this architecture. However, for the ternary quantization, our method beats the state-of-arts TWN with a large margin. By comparing the accuracy of ternary quantization and binary quantization, we find that ternary network works much better than binary network. We also would like to indicate that the ternary network is more computing efficient than binary network because of the existing of zero entries in the weights.

For the one-bit and two-bits shift quantization, we find the the accuracy of the quantized neural network is very close to the full precision network. This implies that the heavy redundance of the float-point network.

Accuracy	Binary	BWN[28]	Ternary	TWN[23]	{-2, +2}	{-4, +4}	Full Precision
Top-1	**0.570**	0.568	**0.582**	0.575	0.592	0.600	0.600
Top-5	**0.797**	0.794	**0.806**	0.798	0.818	0.822	0.824

Table 1: Accuracy of Alexnet on ImageNet classification

Results on ResNet: The results on ResNet-18 are shown in Table 2. ResNet-18 has 18 convolutional layers with shortcut connections. For the proposed method, both the binary and ternary quantization substantially outperforms their competitors on this architecture. For example, our binary network outperforms BWN by 4 points in top-1 accuracy and 3.2 points in top-5 accuracy. The proposed ternary quantization outperforms TWN by 5.2 points and 3.3 points in top-1 and top-5 accuracy respectively, which means a significant gap on ImageNet datasets. We also observe over two percent improvement for ternary quantization than binary quantization.

For ResNet-18, there is a more noticeable gap between full precision network and low-bits quantized networks. For Alexnet, two-bits shift quantization achieves the same top-1 accuracy with full precision network. However, on ResNet-18, we notice 1 point gap between low-bits quantizied and full precision networks. These results suggest that training low-bits quantized network is easier for AlexNet than for ResNet-18, which also implies the parameters in Alexnet are more redundant than those in ResNet-18. This finding is consistent with the finding in other studies such as [17]. In SqueezeNet, the authors find that one can achieve AlexNet-level accuracy on ImageNet with 50x fewer parameters.

Accuracy	Binary	BWN[28]	Ternary	TWN[23]	{-2, +2}	{-4, +4}	Full Precision
Top-1	**0.648**	0.608	**0.670**	0.618	0.675	0.680	0.691
Top-5	**0.862**	0.830	**0.875**	0.842	0.879	0.883	0.890

Table 2: Accuracy of ResNet-18 on ImageNet classification

We also investigate the effectiveness of the proposed method on very deep convolutional network such as ResNet-50. Besides significantly increased network depth, ResNet-50 has a more complex network architecture in comparison to ResNet-18. For instance, a "bottleneck" building block with 1×1 kernel is designed to reduce the computation efforts in ResNet-50. Table 3 shows the comparison results of ResNet-50 on ImageNet. It is easy to observe the similar trends as in ResNet-18. Note that the BWN and TWN papers don't report their results on this network, so we conduct the experiments based on their open

Accuracy	Binary	BWN [28]	Ternary	TWN [23]	{-2, +2}	{-4, +4}	Full Precision
Top-1	**0.687**	0.639	**0.725**	0.656	0.739	0.740	0.753
Top-5	**0.886**	0.851	**0.907**	0.865	0.915	0.916	0.922

Table 3: Accuracy of ResNet-50 on ImageNet classification

source codes.

4.2 Object Detection

4.2.1 Implementation Details

In order to examine the proposed method on object detection task, we apply it to the state-of-art framework SSD [26]. The SSD detection models are trained on Pascal VOC2007 and VOC2012 train dataset, and tested on Pascal VOC 2007 test dataset (4952 images). For SSD, we adopt the open implementation released by the authors on github. For all the experiments, we follow the same setting as in [26] and the input size of input images are 300×300.

The proposed method are evaluated on two base models, i.e., VGG16 and darknet reference model [26]. Both base networks are pre-trained on ImageNet dataset. Note that the VGG16 network [26] here is a variant of original one [30]. In detail, the fc6 and fc7 are converted to convolutional layers with 1×1 kernel, and the fc8 layer is removed. The parameters of fc6 and fc7 are also subsampled. The darknet reference model is borrowed from YOLO [29], which is another fast detection framework. This model is designed to be small yet power, which attains comparable accuracy performance as AlexNet but only with about 10% of the parameters. The model involves 28MBytes parameters. We utilize the base darknet model downloaded from the website.

4.2.2 Results and Analysis

Table 4 shows the mean average precision (mAP) on both models. To the best of our knowledge, there is no other works about low-bits quantization applied their algorithms to the object detection tasks, so we only compare our quantized network with full precision network in this experiment. We only implement two-bits shift quantization. We find that the mAP of both modes are very close to the full precision version. For VGG16+SSD, we only suffer a loss of 0.002 in mAP. These results imply that our proposed method is also effective on the object detection tasks.

Darknet has utilized many 1×1 kernels as in Resnet-50 to accelerate the inference process. In practice 1×1 kernels can be implemented with highly optimized general matrix multiply functions without requiring the reordering in memory (i.e., img2col). We implement two versions of Darknet. In the first version, the 1×1 kernels are also quantized with {-4, +4}, while in the second version, these kernels are not quantized. As shown in Table

mAP	{-4, +4}	Full Precision
Darknet+SSD	0.624 (0.639)	0.642
VGG16+SSD	0.776	0.778

Table 4: mAP of VGG+SSD and Darknet+SSD on Pascal VOC 2007

3, the first version achieves a mAP of 0.624, and the second version obtains a improvement of 1.5 points.

5 Conclusion

This work focuses on model compression and acceleration techniques of deep neural networks. It addresses one fundamental issue with deep learning model, i.e. although deep learning model is highly effective for various detection and classification tasks, its high computational cost makes it difficult to be deployed to scenarios where either memory or computational resources are limited. Inspired from the efficient heuristics proposed to handle MIP problems, we proposed to learn low-bits neural network in the framework of ADMM. Extensive experiments on convolutional neural network for image recognition, object detection, and recurrent neural network for language model have shown the effectiveness of the proposed method.

Reference

[1] A. Alavian and M. C. Rotkowitz, "Improving admm-based optimization of mixed integer objectives," in *Information Sciences and Systems (CISS), 2017 51st Annual Conference on.* IEEE, 2017, pp. 1–6.

[2] 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*, vol. 3, no. 1, pp. 1–122, 2011.

[3] Z. Cheng, D. Soudry, Z. Mao, and Z. Lan, "Training binary multilayer neural networks for image classification using expectation backpropagation," *arXiv preprint arXiv:1503.03562*, 2015.

[4] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *arXiv preprint arXiv:1412.7024*, 2014.

[5] ——, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in Neural Information Processing Systems*, 2015.

[6] 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.

[7] M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[8] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in Neural Information Processing Systems*, 2014.

[9] N. Derbinsky, J. Bento, V. Elser, and J. S. Yedidia, "An improved three-weight message-passing algorithm," *arXiv preprint arXiv:1305.1961*, 2013.

[10] T. Dettmers, "8-bit approximations for parallelism in deep learning," *arXiv preprint arXiv:1511.04561*, 2015.

[11] S. Diamond, R. Takapoui, and S. Boyd, "A general system for heuristic solution of convex problems over nonconvex sets," *arXiv preprint arXiv:1601.07277*, 2016.

[12] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[16] I. Hubara, D. Soudry, and R. E. Yaniv, "Binarized neural networks," *arXiv preprint arXiv:1602.02505*, 2016.

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[20] G. M. Korpelevich, "An extragradient method for finding saddle points and for other problems," *Matecon*, 1976.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012.

[22] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[23] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[24] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," *arXiv preprint arXiv:1511.06393*, 2015.

[25] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," *arXiv preprint arXiv:1510.03009*, 2015.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *European Conference on Computer Vision*, 2016.

[27] T. P. Nguyen, E. Pauwels, E. Richard, and B. W. Suter, "Extragradient method in optimization: Convergence and complexity," *arXiv preprint arXiv:1609.08177*, 2016.

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *European Conference on Computer Vision*, 2016.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[33] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, "A simple effective heuristic for embedded mixed-integer quadratic programming," *International Journal of Control*, pp. 1–11, 2017.

[34] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*, 2015.

[35] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[36] A. K. Yadav, R. Ranjan, U. Mahbub, and M. C. Rotkowitz, "New methods for handling binary constraints," in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*. IEEE, 2016, pp. 1074–1080.

[37] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[38] 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.