SIAM J. IMAGING SCIENCES
Vol. 11, No. 4, pp. 2205–2223

BinaryRelax: A Relaxation Approach for Training Deep Neural Networks with Quantized Weights*

Penghang Yin†, Shuai Zhang‡, Jiancheng Lyu‡, Stanley Osher†, Yingyong Qi‡, and Jack Xin‡

Abstract. We propose BinaryRelax, a simple two-phase algorithm, for training deep neural networks with quantized weights. The set constraint that characterizes the quantization of weights is not imposed until the late stage of training, and a sequence of *pseudo* quantized weights is maintained. Specifically, we relax the hard constraint into a continuous regularizer via a Moreau envelope, which turns out to be the squared Euclidean distance to the set of quantized weights. The pseudo quantized weights are obtained by linearly interpolating between the float weights and their quantizations. A continuation strategy is adopted to push the weights toward the quantized state by gradually increasing the regularization parameter. In the second phase, an exact quantization scheme with a small learning rate is invoked to guarantee fully quantized weights. We test BinaryRelax on the benchmark CIFAR and ImageNet color image datasets to demonstrate the superiority of the relaxed quantization approach and the improved accuracy over the state-of-the-art training methods. Finally, we prove the convergence of BinaryRelax under an approximate orthogonality condition.

Key words. BinaryRelax, deep neural networks, quantization, continuous relaxation

AMS subject classifications. 90C10, 90C26, 90C90

DOI. 10.1137/18M1166134

1. Introduction.

Deep neural networks (DNNs) have achieved remarkable success in computer vision, speech recognition, and natural language processing systems [19, 22, 21, 34]. There is thus a growing interest in deploying DNNs on low-power embedded systems with limited memory storage and computing power, such as cell phones and other battery-powered devices. However, DNNs typically require hundreds of megabytes of memory storage for the trainable full-precision floating-point parameters or weights and need billions of FLOPs to make a single inference. This makes the deployment of DNNs impractical on portable devices. Recent efforts have been devoted to the training of DNNs with coarsely quantized weights which are represented using low-precision (8 bits or less) fixed-point arithmetic [16, 10, 24, 45, 46, 40, 44, 30, 2, 43, 26]. Quantized neural networks enable substantial memory savings and computation/power efficiency while achieving competitive performance with that

*Received by the editors January 22, 2018; accepted for publication (in revised form) July 31, 2018; published electronically October 2, 2018. The first, second, and third authors contributed equally.
http://www.siam.org/journals/siims/11-4/M116613.html
Funding: The work of the authors was partially supported by NSF grants DMS-1522383 and IIS-1632935, ONR grant N00014-16-1-2157, DOE grant DE-SC00183838, and AFOSR grant FA 9550-15-0073.

†Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90095 (yph@ucla.edu, sjo@math.ucla.edu).
‡Department of Mathematics, University of California at Irvine, Irvine, CA 92697 (szhang3@uci.edu, jianchel@uci.edu, yqi@uci.edu, jxin@math.uci.edu).

of full-precision DNNs. Moreover, quantized weights can exploit hardware-friendly bitwise operations and lead to dramatic acceleration at inference time.

The simplest way to perform quantization would be directly rounding the weights of a pretrained full-precision network. But without retraining, this naive approach often leads to poor accuracy at bit-width under 8. From the perspective of optimization, the training of quantized networks can be naturally abstracted as a constrained optimization problem of minimizing some empirical risk subject to a set constraint that characterizes the quantization of weights:

$$\min_{x\in\mathbb{R}^n} f(x) := \frac{1}{N}\sum_{j=1}^{N}\ell_j(x) \quad \text{subject to} \quad x\in\mathcal{Q}. \tag{1}$$

The problem has specific structures. Given a training sample of input I_j and label u_j, the corresponding training loss takes the form

$$\ell_j(x) = \ell(\sigma_l(x_l * \cdots \sigma_1(x_1 * I_j)), u_j),$$

where $x = [x_{(1)}^\top, \ldots, x_{(l)}^\top]^\top$ and $x_{(i)} \in \mathbb{R}^{n_i}$ contains the n_i weights in the ith linear (fully connected or convolutional) layer with $\sum_{i=1}^{l} n_i = n$, σ_i is some elementwise nonlinear function, and "$*$" denotes either matrix-vector product or convolution operation; reshaping is necessary to avoid mismatch in dimensions. For layerwise quantization, the set $\mathcal{Q}$ takes the form of $\mathcal{Q}_1 \times \cdots \times \mathcal{Q}_l$, where $x_{(i)} \in \mathcal{Q}_i := \mathbb{R}_+ \times \{\pm q_1, \pm q_2, \ldots, \pm q_m\}^{n_i}$. Here $\mathbb{R}_+$ denotes the set of nonnegative real numbers and $0 \le q_1 < q_2 < \cdots < q_m$ represent the m quantization levels and are predetermined. The weight vector in the ith layer enjoys the factorization $x_{(i)} = s_i \cdot Q_{(i)}$ for some $Q_{(i)} \in \{\pm q_1, \pm q_2, \ldots, \pm q_m\}^{n_i}$ and some trainable layerwise scalar $s_i \ge 0$. Note that s_i does not have to be low-precision. s_i is shared by all weights across the ith linear layer and will be stored separately from the quantized numbers Q_i for deployment efficiency. The storage for the scaling factors is *negligible* as there are so few of them. Weight quantization has two special cases as follows:

- 1-bit binarization: $m = 1$ and $\mathcal{Q}_i = \mathbb{R}_+ \times \{\pm 1\}^{n_i}$. The storage of $Q_{(i)}$'s only needs 1 bit for representing the signs. Compared to the full-precision model, we have $32\times$ memory savings.
- 2-bit ternarization: $m = 2$ and $\mathcal{Q}_i = \mathbb{R}_+ \times \{0, \pm 1\}^{n_i}$. The storage needs 2 bits for representing the signs and the binary numbers $\{0, 1\}$. Therefore, it gives $16\times$ model compression rate.

The acceleration through low-bit weights is achieved by leveraging the distributive law during forward propagation. For example, propagation through the first linear layer yields the computation of

$$x_{(1)} * I = (s_1 \cdot Q_{(1)}) * I = s_1 \cdot (Q_{(1)} * I).$$

When $Q_{(1)}$ is under 1-bit or 2-bit representation, the computation of $Q_{(1)} * I$ can be extremely fast as there are additions/subtractions involved only.

On the computational side, with sampled mini-batch gradient ∇f_k at the kth iteration, the classical projected stochastic gradient descent (PSGD) [9, 36]

$$\begin{cases} y^{k+1} = x^k - \gamma_k \nabla f_k(x^k), \\ x^{k+1} = \mathrm{proj}_{\mathcal{Q}}(y^{k+1}) \end{cases} \tag{2}$$

performs poorly, however, and gets stagnated when updated with a small learning rate γ_k. It is the quantization/projection of weights that "rounds off" small gradient updates and causes the plateau as explained by Li et al. in a recent study [25]. Instead of using the standard gradient step in (2), a hybrid gradient update

$$y^{k+1} = y^k - \gamma_k \nabla f_k(x^k)$$

was adopted by Courbariaux, Bengio, and David [10] and showed significantly improved accuracy. This modification of PSGD is referred to as BinaryConnect in [25]. BinaryConnect has become the workhorse algorithm for training quantized DNN models such as Xnor-Net [33] and TWN [24]. By introducing the augmented Lagrangian of (1), more complicated algorithms based on alternating minimization were proposed in [4] and [23]. Despite the succinctness and effectiveness of BinaryConnect, its convergence still lacks understanding. The only analysis so far, to our knowledge, appeared in [25] under convexity assumption on the loss function. Researchers have also explored different quantizers, whether uniform or not [10, 33, 45, 24, 40, 30, 44, 5, 20]. All these methods maintain a sequence of purely quantized weights, if not the optimal, during the training.

In this paper, we propose a novel relaxed quantization approach called BinaryRelax to explore more freely the nonconvex landscape of the objective function of the DNNs under the discrete quantization constraint. We relax the set constraint into a continuous regularizer, which leads to a relaxed quantization update. In addition, we set an increasing regularization parameter, driving x^k slowly to the quantized state. When the training error stops decaying at small γ_k, we switch to regular quantization to get genuinely quantized weights as desired. By exploiting the structure of quantization set $\mathcal{Q}$, we prove the convergence of BinaryRelax in the nonconvex setting, which naturally covers that of BinaryConnect.

The rest of the paper is organized as follows. In section 2, we introduce the proposed BinaryRelax method. In section 3, we benchmark CIFAR-10 and CIFAR-100 datasets and compare BinaryRelax with state-of-the-art methods to demonstrate the benefits of performing relaxed quantization. In section 4, we establish the convergence results. Concluding remarks are given in section 5. All technical proofs will be provided in the appendix.

Notation. $\|\cdot\|$ denotes the Euclidean norm; $\|\cdot\|_1$ denotes the ℓ_1 norm; $\|\cdot\|_0$ counts the number of nonzero components. $\mathbf{0} \in \mathbb{R}^n$ represents the vector of zeros. For any vector $x \in \mathbb{R}^n$ and closed set $\mathcal{Q} \subset \mathbb{R}^n$,

$$\mathrm{proj}_{\mathcal{Q}}(x) := \arg\min_{z \in \mathcal{Q}} \|x - z\|$$

is the projection of x onto $\mathcal{Q}$, and

$$\mathrm{dist}(x, \mathcal{Q}) := \min_{z \in \mathcal{Q}} \|x - z\|$$

is the Euclidean distance between x and $\mathcal{Q}$. When $\mathcal{Q}$ is a subspace in $\mathbb{R}^n$, $x \perp \mathcal{Q}$ means that x is orthogonal to $\mathcal{Q}$. $\mathrm{sign}(x)$ is the signum function acting pointwise on x, i.e.,

$$\mathrm{sign}(x)_i := \begin{cases} 1 & \text{if } x_i > 0, \\ -1 & \text{if } x_i < 0, \\ 0 & \text{if } x_i = 0. \end{cases}$$

2. BinaryRelax. Without loss of generality, we assume the set of quantized weights

$$\mathcal{Q} = \mathbb{R}_+ \times \{\pm q_1, \dots, \pm q_m\}^n$$

throughout the paper. With that said, we only consider the case for simplicity that a single adjustable scaling factor is shared by all weights in the network.

2.1. Quantization. In fact, for general b-bit quantization,

$$\mathcal{Q} = \bigcup_{i=1}^{p} \mathcal{L}_i$$

is the union of p distinct one-dimensional subspaces $\mathcal{L}_i \subset \mathbb{R}^n$, $i = 1, 2, \dots, p$, where

$$\mathcal{L}_i = \{s \cdot L_i : s \in \mathbb{R}\}$$

for some $L_i \in \{\pm q_1, \dots, \pm q_m\}^n \setminus \{\mathbf{0}\} \subset \mathbb{R}^n$. See also [23]. Figure 1 shows an example of $\mathcal{Q} = \mathbb{R}_+ \times \{0, \pm 1\}^2$, i.e., the ternarization of two weights.

Given a float weight vector y, its quantization x is basically the projection of y onto the set $\mathcal{Q}$, which gives rise to the optimization problem

$$x = \arg\min_{z \in \mathcal{Q}} \|z - y\|^2 = \mathrm{proj}_{\mathcal{Q}}(y). \tag{3}$$

Note that $\mathcal{Q}$ is a nonconvex set; then the projection may not be unique. In that case, we just assume x is one of them. The above projection/quantization problem can be reformulated as

$$(s^*, Q^*) = \arg\min_{s,Q} \|s \cdot Q - y\|^2 \quad \text{subject to} \quad Q \in \{\pm q_1, \dots, \pm q_m\}^n. \tag{4}$$

The quantization of y is then given by $\mathrm{proj}_{\mathcal{Q}}(y) = s^* \cdot Q^*$. Problem (4) is essentially a constrained K-means clustering problem of one-dimensional points. The centroids are of the form $\pm(s \cdot q_j)$ with $1 \le j \le m$, and they are determined by a single parameter s since q_j's are fixed. For uniform quantization where $q_j = j - 1$, these centroids are equispaced. Given

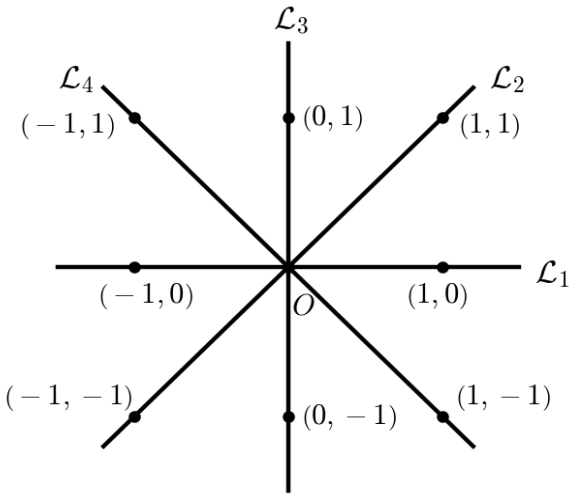

Figure 1. *Graphic illustration of* $\mathcal{Q} = \mathbb{R}_+ \times \{0, \pm 1\}^2$. *In this case,* $b = n = 2$, $p = 4$.

s, the assignment of float weights is then governed by Q. So the problem (4) in principle can be solved by a variant of Lloyd's algorithm [27], which iterates between the assignment step (Q-update) and the centroid update step (s-update). In the Q-update of the lth iteration, fixing the scaling factor s^{l-1}, each Q_i^l is chosen from $\{\pm q_1, \dots, \pm q_m\}$ so that $s^{l-1}Q_i^l$ is the nearest centroid to y_i. In the s-update, a quadratic problem

$$\min_{s\in\mathbb{R}} \ \|s \cdot Q^l - y\|^2$$

is solved by $s^l = \frac{\langle Q^l, y\rangle}{\|Q^l\|^2}$.

The above procedure, however, is impractical here, as quantization is needed in every iteration of training. It has been shown that the closed form (exact) solution of (4) can be computed at $O(n)$ complexity for binarization [33] where $Q \in \{\pm 1\}^n$:

$$s^* = \frac{\|y\|_1}{n}, \ Q_i^* = \begin{cases} 1 & \text{if } y_i \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{5}$$

In the case of ternarization where $Q \in \{0, \pm 1\}^n$, an $O(n \log n)$ exact formula was found in [40]:

$$t^* = \arg\max_{1\leq t\leq n} \frac{\|y_{[t]}\|_1^2}{t}, \ s^* = \frac{\|y_{[t^*]}\|_1}{t^*}, \ Q^* = \operatorname{sign}(y_{[t^*]}), \tag{6}$$

where $y_{[t]} \in \mathbb{R}^n$ keeps the t largest component in magnitude of y, while zeroing out the others. For quantization with wider bit-width ($b > 2$), accurately solving (4) becomes computationally intractable [40]. Empirical formulas have thus been proposed for an approximate quantized solution [24, 40, 44], and they turn out to be sufficient for practical use. For example, a thresholding scheme of $O(n)$ complexity for ternarization was proposed in [24] as

$$\delta = \frac{0.7\|y\|_1}{n}, \ s^* = \frac{\sum_{i=1}^n |y_i| \cdot 1_{|y_i|\geq\delta}}{\sum_{i=1}^n 1_{|y_i|\geq\delta}}, \ Q_i^* = \begin{cases} \operatorname{sign}(y_i) & \text{if } |y_i| \geq \delta, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

For $b > 2$, Yin et al. [40] proposed to just perform one iteration of Lloyd's algorithm with a carefully initialized Q.

The focus of this paper is not on how to quantize a float weight vector. From now on, we simply assume that the quantization $\operatorname{proj}_{\mathcal{Q}}(y)$ can be computed precisely, regardless of the choice of q_j's.

2.2. Moreau envelope and proximal mapping. In the seminal paper [29], Moreau introduced what is now called the Moreau envelope and the proximity operator (also known as proximal mapping) that generalizes the projection. Let $g : \mathbb{R}^n \to (-\infty, \infty]$ be a lower semicontinuous extended-real-valued function. For any $t > 0$, the Moreau envelope function g_t is defined by

$$g_t(x) := \inf_{z\in\mathbb{R}^n} \ g(z) + \frac{1}{2t}\|z - x\|^2.$$

In general, g_t is everywhere finite and locally Lipschitz continuous. Moreover, g_t converges pointwise to g as $t \to 0^+$. The Moreau envelope is closely related to the inviscid Hamilton–Jacobi equation [7]

$$u_t + \frac{1}{2} |\nabla_x u|^2 = 0, \quad u(x,0) = g(x),$$

where $u(x,t) = g_t(x)$ is the unique viscosity solution of the above initial-value problem via the Hopf–Lax formula

$$u(x,t) = \inf_z \left\{ g(z) + tH^* \left(\frac{z-x}{t} \right) \right\}$$

with the Hamiltonian $H(t,x,v) = \frac{1}{2}\|v\|^2$ and its Fenchel conjugate $H^* = H$. The proximal mapping of g is defined by

$$\text{prox}_g(x) := \arg\min_{z \in \mathbb{R}^n} \ g(z) + \frac{1}{2}\|z-x\|^2.$$

It is frequently used in optimization algorithms associated with nonsmooth optimization problems such as total variation denoising [14].

In particular, if $g = \chi_{\mathcal{A}}$ is the indicator function of a close set $\mathcal{A} \subset \mathbb{R}^n$, where

$$\chi_{\mathcal{A}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{A}, \\ \infty & \text{otherwise.} \end{cases}$$

The Moreau envelope is well defined for $t > 0$ and is given by

$$\inf_z \ \chi_{\mathcal{A}}(z) + \frac{1}{2t}\|z-x\|^2 = \inf_{z \in \mathcal{A}} \ \frac{1}{2t}\|z-x\|^2 = \frac{1}{2t}\text{dist}(x,\mathcal{A})^2.$$

The proximal mapping $\text{prox}_g(x)$ reduces to the projection $\text{proj}_{\mathcal{A}}(x)$.

2.3. Relaxed quantization. Let us begin with the alternative form of DNNs quantization problem (1):

$$\min_{x \in \mathbb{R}^n} \ f(x) + \chi_{\mathcal{Q}}(x). \tag{8}$$

When both the objective function $f(x)$ and the set $\mathcal{Q}$ are nonconvex, the discontinuity of $\chi_{\mathcal{Q}}$ poses an extra challenge in minimization since a continuous gradient descent update can be made stagnant when projected discontinuously. The Moreau envelope of $\chi_{\mathcal{Q}}$ is $\frac{1}{2t}\text{dist}(x,\mathcal{Q})^2$, which is continuously differentiable almost everywhere, except at points that have at least two nearest line subspaces, i.e., there exist two different ways to quantize x. We use $\frac{1}{2t}\text{dist}(x,\mathcal{Q})^2$ as the approximant of the discontinuous $\chi_{\mathcal{Q}}(z)$ and propose to minimize the relaxed training error

$$\min_{x \in \mathbb{R}^n} \ f(x) + \frac{\lambda}{2}\text{dist}(x,\mathcal{Q})^2, \tag{9}$$

where $\lambda = t^{-1} > 0$ is the regularization parameter. When $\lambda \to \infty$, $\frac{\lambda}{2}\text{dist}(x,\mathcal{Q})^2$ converges pointwise to $\chi_{\mathcal{Q}}(x)$, and the global minimum of (9) converges to that of (8).

Proposition 2.1. *Suppose $f(x)$ is continuous. Let $f_{\mathcal{Q}}^* = \min_{x\in\mathcal{Q}} f(x)$ be the global minimum of (8) and x_λ^* be the global minimizer of relaxed quantization problem (9). Then*

$$\operatorname{dist}(x_\lambda^*, \mathcal{Q}) \to 0 \text{ and } f(x_\lambda^*) \to f_{\mathcal{Q}}^* \text{ as } \lambda \to \infty.$$

Remark 2.2. Relaxation via the Moreau envelope leads to a quadratic penalty formulation. An interesting but different quadratic penalty was considered in [4] for the general compression problem of DNNs. In fact, by replacing the Euclidean distance $\|\cdot\|$ in the Moreau envelope with the metric $\|\cdot\|_D$ induced by some matrix D, one can derive a more general penalty

$$\inf_{z\in\mathcal{A}} \frac{1}{2}\|z - x\|_D^2.$$

We refer the readers to the recent paper [32] for successful application of such a penalty to the phase retrieval problem.

2.4. Algorithm. Inspired by the hybrid gradient update proposed in [10], we write a two-line solver for the minimization problem (9):

$$\begin{cases} y^{k+1} = y^k - \gamma_k \nabla f_k(x^k), \\ x^{k+1} = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|x - y^{k+1}\|^2 + \frac{\lambda}{2}\operatorname{dist}(x, \mathcal{Q})^2. \end{cases} \tag{10}$$

The algorithm constructs two sequences: an auxiliary sequence of float weights $\{y^k\}$ and a sequence of *nearly* quantized weights $\{x^k\}$. The mismatch of discontinuous projection and continuous gradient descent is resolved by the relaxed quantization step in (10), which calls for computing the proximal mapping of the function $\frac{\lambda}{2}\operatorname{dist}(x, \mathcal{Q})^2$. This can be done via the following closed-form formula.

Proposition 2.3. *Let*

$$\operatorname{proj}_{\mathcal{Q}}(y^{k+1}) = \arg\min_{x\in\mathcal{Q}} \|x - y^{k+1}\|^2$$

be the quantization of y^{k+1}; then the solution to the relaxed quantization subproblem in (10) is given by

$$x^{k+1} = \frac{\lambda \operatorname{proj}_{\mathcal{Q}}(y^{k+1}) + y^{k+1}}{\lambda + 1}. \tag{11}$$

Note that we still need the exact quantization $\operatorname{proj}_{\mathcal{Q}}(y^{k+1})$ to perform relaxed quantization. The update x^{k+1} is essentially a linear interpolation between y^{k+1} and its quantization $\operatorname{proj}_{\mathcal{Q}}(y^{k+1})$, and λ controls the weighted average. x^{k+1} is thus not quantized because $x^{k+1} \notin \mathcal{Q}$, but x^{k+1} approaches $\mathcal{Q}$ as λ increases. Hereby we adopt a continuation strategy and let λ grow slowly. Specifically, we inflate λ after a certain number of epochs by a factor $\rho > 1$. Intuitively, the relaxation with continuation will help skip over some bad local minima of (8) located in $\mathcal{Q}$, because they are not local minima of the relaxed formulation in general.

Proposition 2.4. *Suppose $f(x)$ is differentiable. Any point $x^* \in \mathcal{Q}$ is not a local minimizer of the relaxed quantization problem (9) unless $\nabla f(x^*) = \mathbf{0}$.*

Algorithm 1. BinaryRelax.

Input: number of epochs for training, batch size, schedule of learning rate $\{\gamma_k\}$, growth factor $\rho > 1$

```
for i = 1, 2,..., nb-epoch do
  Randomly shuffle the data and partition into batches.
  for j = 1, 2, ..., nb-batch do
    y^{k+1} = y^k − γ_k ∇f_k(x^k)
    if i ≤ T then
      x^{k+1} = (λ_k proj_Q(y^{k+1}) + y^{k+1}) / (λ_k + 1)    // Phase I
      if increase λ then
        λ_{k+1} = ρ λ_k
      else
        λ_{k+1} = λ_k
      end if
    else
      x^{k+1} = proj_Q(y^{k+1})    // Phase II
    end if
    k = k + 1
  end for
end for
```

In order to obtain quantized weights in the end, we turn off the relaxation mode and enforce quantization. The BinaryRelax algorithm is summarized in Algorithm 1.

Remark 2.5. For BinaryRelax, we replace a discrete quantization constraint with a continuous regularizer. The similar idea of relaxing the discrete sparsity constraint $\|x\|_0 \le s$ into a continuous and possibly nonconvex sparse regularizer has been long known in the contexts of statistics and compressed sensing [38, 13, 3]. For example, compressed sensing solvers for minimizing the convex ℓ_1 norm [14] or nonconvex sparse proxies, such as $\ell_{1/2}$ (with smoothing) [6] and ℓ_{1-2} [39], often empirically outperform those directly tackling the nonzero counting metric ℓ_0. Interestingly, similar to the quantization set $\mathcal{Q}$, the sparsity constraint set $\{x \in \mathbb{R}^n : \|x\|_0 \le s\}$ is also a finite union of low-dimensional subspaces in $\mathbb{R}^n$. More precisely,

$$\{x \in \mathbb{R}^n : \|x\|_0 \le s\} = \bigcup_{\mathcal{S}\subset\{1,\dots,n\},\, |\mathcal{S}|=s} \{x \in \mathbb{R}^n : \operatorname{supp}(x) \subseteq \mathcal{S}\},$$

where $\operatorname{supp}(x)$ denotes the support of x, and each member in the union is an s-dimensional subspace with $s \ll n$.

Remark 2.6. BinaryRelax resembles the linearized Bregman algorithm proposed by Yin et al. [41, 42] for solving the basis pursuit problem [8, 3]

$$\min_{x\in\mathbb{R}^n} \|x\|_1 \quad \text{subject to} \quad Ax = b,$$

by iterating

$$\begin{cases} y^{k+1} = y^k - \tau_k A^\top (Ax^k - b), \\ x^{k+1} = \delta \cdot \text{shrink}(y^{k+1}, \mu), \end{cases}$$

where δ, μ, $\tau_k > 0$ are algorithmic parameters. In linearized Bregman, $A^\top(Ax - b)$ is the gradient of sum of squares error $\frac{1}{2}\|Ax - b\|^2$, and $\text{shrink}(y, \mu)$ is the proximal mapping of ℓ_1 norm (also known as the soft-thresholding operator [12]):

$$\text{shrink}(y, \mu) := \arg\min_u \ \frac{1}{2\mu}\|u - y\|^2 + \|u\|_1.$$

With that said, linearized Bregman also iterates between some sort of hybrid gradient step and proximal mapping. However, it is not exactly the same as BinaryRelax, as there is a scaling by δ in the proximal step.

2.5. Connection to BinaryConnect. In fact, Phase II of BinaryRelax,

$$\begin{cases} y^{k+1} = y^k - \gamma_k \nabla f_k(x^k), \\ x^{k+1} = \text{proj}_{\mathcal{Q}}(y^{k+1}), \end{cases} \tag{12}$$

is exactly the BinaryConnect scheme [10]. The performance of BinaryRelax, however, mostly relies on Phase I training. As will be seen from the experimental results reported in section 3, the gain from Phase II training is very limited. Switching to BinaryConnect in Phase II is just to get truly quantized weights.

3. Experimental results. We tested BinaryRelax on the benchmark CIFAR [18] and ImageNet [11] color image datasets. The two baselines are the BinaryConnect framework (12) combined with the exact binarization formula (5) (BWN) [33] and the heuristic ternarization scheme (7) (TWN) [24], resp. We used the same quantization formulas for BinaryRelax in the relaxed quantization update (11). Both algorithms were initialized with the weights of a pretrained float model.

3.1. The selection of λ. We always initialize the relaxation parameter $\lambda_0 = 1$. We split into roughly 4/5 and 1/5 of the training epochs for Phase I and Phase II, resp. To guarantee smooth transitioning to Phase II from Phase I, a proper growth factor $\rho > 1$ is chosen so that $\lambda \in (100, 200)$ at the moment Phase I ends. A relatively small λ will result in a noticeable drop in accuracy when Phase II starts.

3.2. CIFAR datasets. The CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 test images per class. Figure 2 shows some sample images from CIFAR datasets. In the experiments, we used the testing images for validation. We coded up the BinaryRelax in the PyTorch [31] platform. The experiments were carried out on two desktops with Nvidia graphics cards GTX 1080 Ti and Titan X, resp.

We ran 300 epochs. The initial learning rate $\gamma_0 = 0.1$ with decay by a factor of 0.1 at epochs $\{120, 220\}$. Phase II starts at epoch 240. λ increases by a factor of $\rho = 1.02$ after every

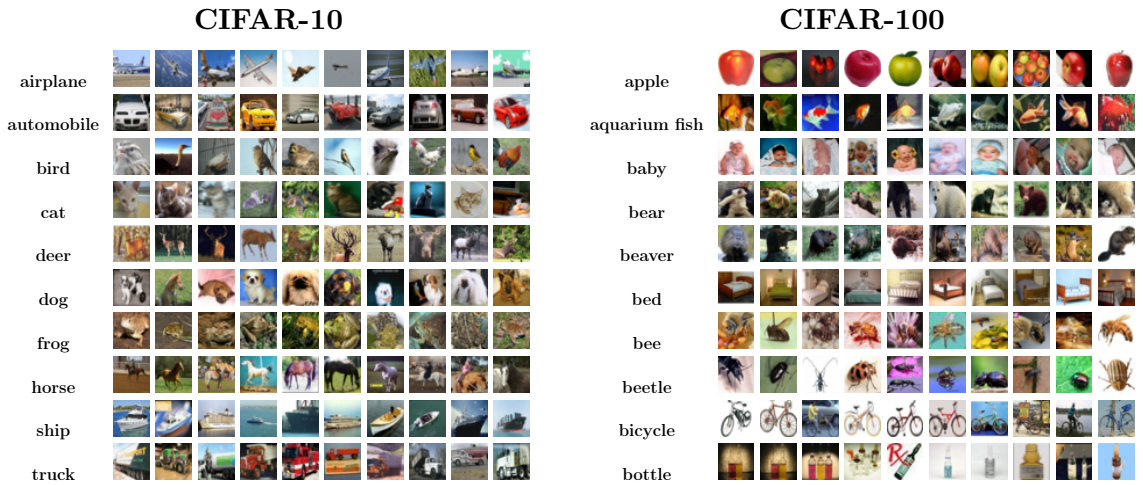

Figure 2. *Sample images from CIFAR datasets:* 10 *classes in CIFAR*-10 *(left);* 10 *out of* 100 *classes in CIFAR*-100 *(right).*

Table 1
CIFAR-10 *validation accuracies.*

CIFAR-10	Float	Binary		Ternary	
		BWN	Ours	TWN	Ours
VGG-11	91.93	88.70	**89.28**	90.48	**91.01**
VGG-16	93.59	91.60	**91.98**	92.75	**93.20**
ResNet-20	92.68	87.44	**87.82**	88.65	**90.07**
ResNet-32	93.40	89.49	**90.65**	90.94	**92.04**
ResNet-18	95.49	92.72	**94.19**	93.55	**94.98**
ResNet-34	95.70	93.25	**94.66**	94.05	**95.07**

Table 2
CIFAR-100 *validation accuracies.*

CIFAR-100	Float	Binary		Ternary	
		BWN	Ours	TWN	Ours
VGG-11	70.43	62.35	**63.82**	64.16	**65.87**
VGG-16	73.55	69.03	**70.14**	71.41	**72.10**
ResNet-56	70.86	66.73	**67.65**	68.26	**69.83**
ResNet-110	73.21	68.67	**69.85**	68.95	**72.32**
ResNet-18	76.32	72.31	**74.04**	73.15	**75.24**
ResNet-34	77.23	72.92	**75.62**	74.43	**76.16**

epoch. In addition, we used batch size = 128, ℓ_2 weight decay = 10^{-4}, batch normalization [17], and momentum = 0.95.

We tested the algorithms on the popular VGG [37] and ResNet[15] architectures, and the validation accuracies for CIFAR-10 and CIFAR-100 are summarized in Tables 1 and 2, resp. Note that ResNet-18 and ResNet-34 tested here were originally constructed for the more challenging ImageNet classification [11] and then adapted for CIFAR datasets. They have wider

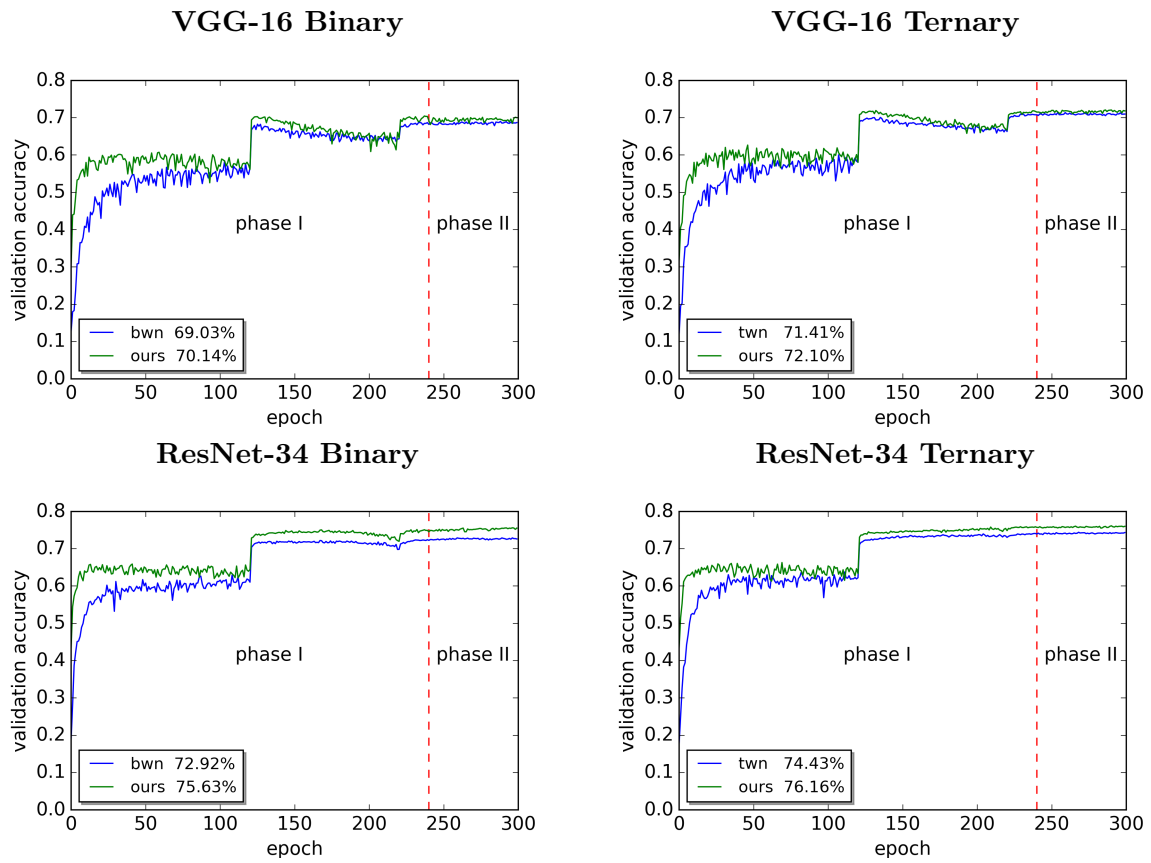

Figure 3. *Comparisons of validation accuracy curves for CIFAR-100 using VGG-16 and ResNet-34. The initial learning rate $\gamma_0 = 0.1$ and decays by a factor of 0.1 at epoch 120 and 220. The initial regularization parameter $\lambda_0 = 1$ and grows by a factor of $\rho = 1.02$ after each epoch until epoch 240, where Phase II starts.*

channels in the convolutional layers and are much larger than the other ResNets. For example, ResNet-18 has ∼11 million parameters, whereas ResNet-110 has only ∼1.7 million. This explains their higher accuracies. All quantized networks were initialized from their full-precision counterparts whose validation accuracies are listed in the second column. Figure 3 shows the validation accuracies for CIFAR-100 tests with VGG-16 and ResNet-34 during the training process. For the VGG-16 tests, we notice the decay of the validation accuracies of BinaryRelax occurs in Phase I training. This is due to the increase of the parameter λ, which makes the regularization more and more stringent. With approximately the same training cost, our relaxed quantization approach consistently outperforms the hard quantization counterpart in validation accuracies. As seen from the tables and figure, the advantage of relaxed quantization is particularly clear when it comes to the large nets ResNet-18 and ResNet-34, where we have more complex landscapes with spurious local minima. In this case, our accuracies of binarized networks even surpass that of TWN. The relaxation indeed helps skip over bad local minima during the training.

Table 3
ImageNet validation accuracies.

Network	Bit-width	Method	Top-1	Top-5
ResNet-18	32 (float)		69.6	89.0
	1 (binary)	BWN	60.8	83.0
		Ours	**63.2**	**85.1**
	2 (ternary)	TWN	61.8	84.2
		Ours	**66.5**	**87.3**

3.3. ImageNet. The ImageNet (ILSVRC12) dataset [11] is a benchmark for the large-scale image classification task, which has 1.2 million images for training and 50,000 for validation of 1,000 categories. We quantize ResNet-18 at bit-widths 1 (binary) and 2 (ternary). The experiments were carried out on a machine with 8 Nvidia GeForce GTX 1080 Ti GPUs.

We initialized BinaryRelax with the pretrained full-precision (32-bit) models available from the PyTorch torchvision package [31]. We trained in total 70 epochs, with Phase II starting at epoch 55. The initial learning rate $\gamma_0 = 0.1$ and decays by a factor of 0.1 at epochs $\{30, 40, 50\}$. Relaxation parameter λ starts at 1 and increases by a growth factor of $\rho = 1.045$ after each half (1/2) epoch. In all these experiments, we used momentum= 0.9 and weight decay $= 10^{-4}$. The comparison results with BWN and TWN are listed in Table 3.

4. Convergence analysis. In this section, we analyze the convergence property of the proposed BinaryRelax. More precisely, we will focus on Phase II of BinaryRelax (i.e., Binary Connect):

$$\begin{cases} y^{k+1} = y^k - \gamma_k \nabla f_k(x^k), \\ x^{k+1} = \mathrm{proj}_{\mathcal{Q}}(y^{k+1}). \end{cases} \tag{13}$$

Although the convergence of BinaryConnect at a small learning rate is observed empirically (as seen from our experiments), the only convergence results, to our knowledge, were proved in [25], in terms of the objective value of float ergodic averages $\{f(\frac{\sum_{i=1}^k y^i}{k})\}$ under the convexity assumption. Moreover, the quantization set $\mathcal{Q}$ considered in [25] is quite different. They constrain the quantized weights to $\{0, \pm\Delta, \pm 2\Delta, \dots\}$, where $\Delta > 0$ is some fixed resolution. In this case, $\mathcal{Q}$ is an unbounded Δ-lattice in $\mathbb{R}^n$. Recall from section 2.1 that our $\mathcal{Q}$ takes the form of $\cup_{i=1}^p \mathcal{L}_i$ with each $\mathcal{L}_i$ being a line passing through the origin. This assumption on $\mathcal{Q}$ generalizes the binary and ternary cases in the existing literature such as [33, 24]. Without assuming the convexity of f, we will show the sequence $\{x^k\}$ generated by the iteration (13) subsequentially converges in expectation to an approximate critical point. To establish the convergence, we need to exploit the property of the set $\mathcal{Q}$ being the union of line subspaces by introducing several technical lemmata in section 4.1. The analysis here cannot be readily extended to the setup of [25] or other problems under general discrete constraint.

4.1. Preliminaries. We have the following basic assumptions.

Assumption 4.1. $f(x)$ is bounded from below. Without loss of generality, we assume the lower bound is 0.

Assumption 4.2. $f(x)$ is L-Lipschitz differentiable, i.e., for any $x, y \in \mathbb{R}^n$, we have

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|.$$

Assumption 4.3. $\mathbb{E}[\|\nabla f(x^k) - \nabla f_k(x^k)\|^2] \le \sigma^2$ for all $k \in \mathbb{N}$, where the expectation is taken over the stochasticity of the algorithm (i.e., random selection of f_k).

Our proof relies on the following technical lemmata that exploit the structure of set $\mathcal{Q}$.

Lemma 4.4 (approximate orthogonality). *Let $\{y^k\}, \{x^k\}$ be defined in (13). There exists $\alpha_k \ge 0$ such that*

$$\alpha_k\|x^{k+1} - x^k\|^2 + \|y^k - x^k\|^2 = \|y^k - x^{k+1}\|^2.$$

Proposition 4.5. *Let $\theta_{\min}$ be the smallest angle formed by any two line subspaces in $\mathcal{Q}$. If $\|x^{k+1} - x^k\| < \|x^k\| \sin\theta_{\min}$, then $\alpha_k = 1$ in Lemma 4.4. Moreover, α_k may have to be 0 only when $\|y^k - x^k\| = \|y^k - x^{k+1}\|$ and $\nabla f_k(x^k) \perp \mathcal{L}_i$ with $\mathcal{L}_i$ containing x^{k+1}.*

The above proposition implies that α_k is generally positive and approaches 1 when the relative change in consecutive iterates is getting small.

Lemma 4.6 (alternative update). *Let $\{x^k\}$ be defined in (13). Suppose $x^{k+1} \in \mathcal{L}_i \subset \mathcal{Q}$ with $\mathcal{L}_i$ being some line subspace and define $\tilde{x}^k := \mathrm{proj}_{\mathcal{L}_i}(y^k)$; then*

$$x^{k+1} = \arg\min_{x \in \mathcal{L}_i} \|x - (\tilde{x}^k - \gamma_k \nabla f_k(x^k))\|^2.$$

Moreover, x^{k+1} is a local minimizer of the following problem:

$$\min_{x \in \mathcal{Q}} \|x - (\tilde{x}^k - \gamma_k \nabla f_k(x^k))\|^2. \tag{14}$$

Lemma 4.7. *Let α_k and $\tilde{x}^k$ be defined in Lemmata 4.4 and 4.6, resp.; it holds that*

$$\|x^{k+1} - \tilde{x}^k\|^2 \le \alpha_k\|x^{k+1} - x^k\|^2.$$

The following descent lemma will be useful.

Lemma 4.8 (descent lemma [1]). *For any x, y, it holds that*

$$f(x) \le f(y) + \langle \nabla f(y), x - y\rangle + \frac{L}{2}\|x - y\|^2.$$

We recall the definition of subdifferential for proper and lower semicontinuous functions.

Definition 4.9 (subdifferential [28, 35]). *Let $h : \mathbb{R}^n \to (-\infty, +\infty]$ be a proper and lower semicontinuous function. We define $\mathrm{dom}(h) := \{x \in \mathbb{R}^n : h(x) < +\infty\}$. For a given $x \in \mathrm{dom}(h)$, the Fréchet subdifferential of h at x, written as $\hat{\partial} h(x)$, is the set of all vectors $u \in \mathbb{R}^n$ which satisfy*

$$\liminf_{y \ne x,\, y \to x} \frac{h(y) - h(x) - \langle u, y - x\rangle}{\|y - x\|} \ge 0.$$

When $x \notin \mathrm{dom}(h)$, we set $\hat{\partial} h(x) = \emptyset$. The (limiting) subdifferential, or simply the subdifferential, of h at $x \in \mathbb{R}^n$, written as $\partial h(x)$, is defined through the following closure process:

$$\partial h(x) := \{u \in \mathbb{R}^n : \exists\, x^k \to x,\ h(x^k) \to h(x) \text{ and } u^k \in \hat{\partial} h(x^k) \to u \text{ as } k \to \infty\}.$$

4.2. Main results. We are in position to present the convergence results, which are established under an approximate orthogonality condition on α_k in Lemma 4.4.

Theorem 4.10. *Let $\{x^k\}$ be the sequence generated by (13). Suppose there exist $\underline{\alpha}, \bar{\alpha}, \gamma > 0$ such that $\underline{\alpha} \le \alpha_k \le \bar{\alpha}$ and $\gamma_{k+1} \le \gamma_k \le \gamma < \frac{\underline{\alpha}}{2L}$ for all $k \in \mathbb{N}$. Then*

$$\lim_{k\to\infty} \mathbb{E}[\|x^{k+1} - x^k\|^2] = 0$$

if $\sum_{k=0}^{\infty} \gamma_k^2 < \infty$. If further $\sum_{k=0}^{\infty} \gamma_k = \infty$, we have

$$\liminf_{k\to\infty} \mathbb{E}[\operatorname{dist}(\mathbf{0}, \partial h(x^k))^2] \le 3\sigma^2 \left(\frac{4\bar{\alpha}}{\underline{\alpha}^2} + 1\right),$$

where $h = f + \chi_{\mathcal{Q}}$ is the overall objective function.

5. Concluding remarks. From optimization point of view, we proposed BinaryRelax, a novel relaxation approach for training quantized neural networks. Our algorithm iterates between a hybrid gradient step for updating the float weights and a weighted average of the computed float weights and their quantizations. We increase slowly the parameter that controls the average to drive the weights to the quantized state. In order to get the purely quantized weights, exact quantization replaces the weighted average in the second phase of training. Extensive experiments shows that with about the same training cost, BinaryRelax is consistently better than its BinaryConnect counterpart in terms of validation accuracy. It has clearer advantage on larger networks, which yield more complex landscape of the training loss with spurious local minima. In addition, BinaryRelax is provably convergent in expectation under an approximate orthogonality condition, which is another contribution of this paper.

Appendix: Technical proofs.

Proof of Proposition 2.1. Since x_λ^* is the global minimizer of (9),

$$f_{\mathcal{Q}}^* \ge f(x_\lambda^*) + \frac{\lambda}{2}\operatorname{dist}(x_\lambda^*, \mathcal{Q})^2 \ge f^* + \frac{\lambda}{2}\operatorname{dist}(x_\lambda^*, \mathcal{Q})^2,$$

where $f^* = \min_{x\in\mathbb{R}^n} f(x) > -\infty$. So

$$\operatorname{dist}(x_\lambda^*, \mathcal{Q}) \le \sqrt{\frac{2(f_{\mathcal{Q}}^* - f^*)}{\lambda}} \to 0, \text{ as } \lambda \to \infty.$$

Denote $x_{\lambda,\mathcal{Q}}^* = \operatorname{proj}_{\mathcal{Q}}(x_\lambda^*)$; then $\|x_{\lambda,\mathcal{Q}}^* - x_\lambda^*\| \to 0$ as $\lambda \to \infty$. Since $f_{\mathcal{Q}}^*$ is the minimum in $\mathcal{Q}$, further we have

$$f(x_\lambda^*) + \frac{\lambda}{2}\operatorname{dist}(x_\lambda^*, \mathcal{Q})^2 \le f_{\mathcal{Q}}^* \le f(x_{\lambda,\mathcal{Q}}^*) \to f(x_\lambda^*), \text{ as } \lambda \to \infty.$$

Therefore, $\lim_{\lambda\to\infty} f(x_\lambda^*) = f_{\mathcal{Q}}^*$. ∎

Proof of Proposition 2.3. Problem (10) amounts to

$$\min_x \min_{z\in\mathcal{Q}} \frac{1}{2}\|x - y^k\|^2 + \frac{\lambda}{2}\|z - x\|^2 = \min_{z\in\mathcal{Q}} \min_x \frac{1}{2}\|x - y^k\|^2 + \frac{\lambda}{2}\|z - x\|^2.$$

With fixed $z \in \mathcal{Q}$, the inner problem is minimized at $x = \frac{\lambda z + y^k}{\lambda+1}$. Then it reduces to

$$\begin{aligned} z^* &= \arg\min_{z\in\mathcal{Q}} \frac{1}{2}\left\|\frac{\lambda z + y^k}{\lambda+1} - y^k\right\|^2 + \frac{\lambda}{2}\left\|z - \frac{\lambda z + y^k}{\lambda+1}\right\|^2 \\ &= \arg\min_{z\in\mathcal{Q}} \|z - y^k\|^2 = \mathrm{proj}_{\mathcal{Q}}(y^k). \end{aligned}$$

Therefore, $x^k = \frac{\lambda\,\mathrm{proj}_{\mathcal{Q}}(y^k)+y^k}{\lambda+1}$ is the optimal solution. ∎

Proof of Proposition 2.4. Proof by contradiction. Let us assume $x^* \in \mathcal{Q}$ is a local minimizer of problem (9) and $\nabla f(x^*) \neq \mathbf{0}$. Then for any point x in the neighborhood of x^*, we have

$$f(x^*) \le f(x) + \frac{\lambda}{2}\mathrm{dist}(x,\mathcal{Q})^2 \le f(x) + \frac{\lambda}{2}\|x - x^*\|^2.$$

Set $x = x^* - \beta\nabla f(x^*)$ with a small $\beta > 0$. The above inequality reduces to

$$f(x^*) \le f(x^* - \beta\nabla f(x^*)) + \frac{\lambda\beta^2}{2}\|\nabla f(x^*)\|^2. \tag{15}$$

On the other hand, by Taylor's expansion,

$$f(x^* - \beta\nabla f(x^*)) = f(x^*) - \beta\|\nabla f(x^*)\|^2 + o(\beta). \tag{16}$$

Combining (15) and (16), we have

$$\|\nabla f(x^*)\|^2 \le \frac{\lambda\beta}{2}\|\nabla f(x^*)\|^2 + o(1),$$

which leads to a contradiction as we let $\beta \to 0$. ∎

Proof of Lemma 4.4. Since $x^k, x^{k+1} \in \mathcal{Q}$ and $x^k = \mathrm{proj}_{\mathcal{Q}}(y^k)$, it holds that $\|y^k - x^k\|^2 \le \|y^k - x^{k+1}\|^2$, i.e., $\alpha_k \ge 0$. ∎

Proof of Proposition 4.5. Since the only intersection of the line subspaces is the origin, the distance between x^k and any other line is at least $\|x^k\|\sin\theta_{\min}$. If $\|x^{k+1} - x^k\| < \|x^k\|\sin\theta_{\min}$, then x^k and x^{k+1} must lie in the same line, and therefore $\alpha_k = 1$. On the other hand, if α_k can only be 0, then it must hold that $\|y^k - x^k\| = \|y^k - x^{k+1}\|$ and $x^k \neq x^{k+1}$, meaning that x^{k+1} is a different projection of y^k onto $\mathcal{Q}$. Moreover, since the projection of $y^{k+1} = y^k - \gamma_k\nabla f_k(x^k)$ onto $\mathcal{Q}$ is also x^{k+1}. Suppose $x^{k+1} \in \mathcal{L}_i \subset \mathcal{Q}$; then $\nabla f_k(x^k) \perp \mathcal{L}_i$. ∎

Proof of Lemma 4.6. By the assumption, we have

$$x^{k+1} = \mathrm{proj}_{\mathcal{L}_i}\left(y^k - \gamma_k\nabla f_k(x^k)\right) = \mathrm{proj}_{\mathcal{L}_i}\left(\tilde{x}^k - \gamma_k\nabla f_k(x^k) + y^k - \tilde{x}^k\right).$$

Note that $y^k - \tilde{x}^k \perp \mathcal{L}_i$ (see Figure 4); then

$$x^{k+1} = \mathrm{proj}_{\mathcal{L}_i}(\tilde{x}^k - \gamma_k\nabla f_k(x^k)).$$

So x^{k+1} is the closest point to $\tilde{x}^k - \gamma_k\nabla f_k(x^k)$ on $\mathcal{L}_i$. If $\tilde{x}^k - \gamma_k\nabla f_k(x^k) = \mathbf{0}$, then $x^{k+1} = \mathbf{0}$ is the global minimizer of (14). Otherwise, $x^{k+1} \neq \mathbf{0}$. Since the line subspaces that constitute $\mathcal{Q}$ only intersect at the origin, there exists a neighborhood $\mathcal{N}$ of x^{k+1} such that $\mathcal{N} \cap \mathcal{Q} \subset \mathcal{L}_i$. Therefore, x^{k+1} is a local minimizer of problem (14). ∎

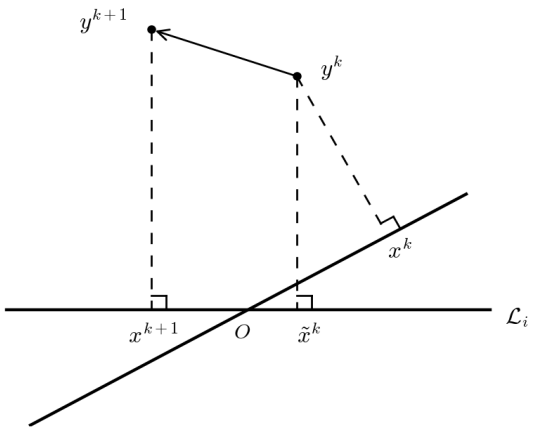

Figure 4. *Illustration of Lemma 4.6.* $y^{k+1} = y^k - \gamma_k \nabla f_k(x^k)$, *so* x^{k+1} *is also the projection of* $\tilde{x}^k - \gamma_k \nabla f_k(x^k)$ *onto* $\mathcal{L}_i$.

Proof of Lemma 4.7. Using the facts $x^k = \mathrm{proj}_{\mathcal{Q}}(y^k)$, $\tilde{x}^k = \mathrm{proj}_{\mathcal{L}_i}(y^k) \in \mathcal{Q}$, and $x^{k+1} \in \mathcal{L}_i$ and invoking Lemma 4.4, we have

$$\begin{aligned}\|x^{k+1} - \tilde{x}^k\|^2 &= \|y^k - x^{k+1}\|^2 - \|y^k - \tilde{x}^k\|^2 \\ &\le \|y^k - x^{k+1}\|^2 - \|y^k - x^k\|^2 = \alpha_k \|x^{k+1} - x^k\|^2.\end{aligned}$$

■

Proof of Theorem 4.10. By Lemma 4.8,

$$\begin{aligned} f(x^{k+1}) &\le f(x^k) + \left\langle \nabla f(x^k), x^{k+1} - x^k \right\rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2 \\ &= f(x^k) + \left\langle \nabla f_k(x^k), x^{k+1} - x^k \right\rangle + \left\langle \nabla f(x^k) - \nabla f_k(x^k), x^{k+1} - x^k \right\rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2. \end{aligned} \tag{17}$$

The cross terms need care. We rewrite the update $x^{k+1} = \mathrm{proj}_{\mathcal{Q}}(y^k - \gamma_k \nabla f_k(x^k))$ as

$$x^{k+1} = \arg\min_{x \in \mathcal{Q}} \langle \nabla f_k(x^k), x \rangle + \frac{1}{2\gamma_k}\|x - y^k\|^2.$$

Since $x^k \in \mathcal{Q}$, we have

$$\left\langle \nabla f_k(x^k), x^{k+1} \right\rangle + \frac{1}{2\gamma_k}\|x^{k+1} - y^k\|^2 \le \left\langle \nabla f_k(x^k), x^k \right\rangle + \frac{1}{2\gamma_k}\|x^k - y^k\|^2.$$

Then by Lemma 4.4,

$$\left\langle \nabla f_k(x^k), x^{k+1} - x^k \right\rangle \le \frac{1}{2\gamma_k}(\|x^k - y^k\|^2 - \|x^{k+1} - y^k\|^2) \le -\frac{\alpha}{2\gamma_k}\|x^{k+1} - x^k\|^2. \tag{18}$$

By Young's inequality,

$$\left\langle \nabla f(x^k) - \nabla f_k(x^k), x^{k+1} - x^k \right\rangle \le \frac{\gamma_k}{\alpha}\|\nabla f(x^k) - \nabla f_k(x^k)\|^2 + \frac{\alpha}{4\gamma_k}\|x^{k+1} - x^k\|^2. \tag{19}$$

Combining (17), (18), and (19) and taking the expectation gives

$$\mathbb{E}[f(x^{k+1})] \le \mathbb{E}[f(x^k)] - \frac{\underline{\alpha} - 2\gamma_k L}{4\gamma_k}\mathbb{E}[\|x^{k+1} - x^k\|^2] + \frac{\gamma_k \sigma^2}{\underline{\alpha}}. \tag{20}$$

Multiplying (20) by γ_k and using $\alpha_k \ge \underline{\alpha} > 0$, $\gamma_{k+1} \le \gamma_k \le \gamma < \frac{\underline{\alpha}}{2L}$, and $f \ge 0$, we obtain

$$\gamma_{k+1}E[f(x^{k+1})] \le \gamma_k \mathbb{E}[f(x^{k+1})] \le \gamma_k \mathbb{E}[f(x^k)] - (\underline{\alpha} - 2\gamma L)\mathbb{E}[\|x^{k+1} - x^k\|^2] + \frac{\gamma_k^2 \sigma^2}{\underline{\alpha}}.$$

Rearranging terms in the above inequality and taking the sum over k, we have

$$(\underline{\alpha} - 2\gamma L)\sum_{k=0}^{\infty} \mathbb{E}[\|x^{k+1} - x^k\|^2] \le \gamma f(x^0) - \lim_{k\to\infty} \gamma_k \mathbb{E}[f(x^k)] + \frac{\sigma^2}{\underline{\alpha}}\sum_{k=0}^{\infty}\gamma_k^2 < \infty.$$

Therefore, $\lim_{k\to\infty} \mathbb{E}[\|x^{k+1} - x^k\|^2] = 0$.

Next we prove the second claim. By Lemma 4.6, the first-order optimality condition of (14) holds at x^{k+1}. So

$$\mathbf{0} \in \nabla f_k(x^k) + \frac{x^{k+1} - \tilde{x}^k}{\gamma_k} + \partial \chi_{\mathcal{Q}}(x^{k+1}),$$

which implies

$$-\frac{x^{k+1} - \tilde{x}^k}{\gamma_k} - \nabla f_k(x^k) + \nabla f(x^{k+1}) \in \nabla f(x^{k+1}) + \partial \chi_{\mathcal{Q}}(x^{k+1}) = \partial h(x^{k+1}).$$

Therefore,

$$\begin{aligned}
&\mathbb{E}[\operatorname{dist}(\mathbf{0}, \partial h(x^{k+1}))^2] \\
&\le \mathbb{E}\left[\left\| -\frac{x^{k+1} - \tilde{x}^k}{\gamma_k} - \nabla f_k(x^k) + \nabla f(x^{k+1})\right\|^2\right] \\
&\le 3\left(\mathbb{E}\left[\frac{\|x^{k+1} - \tilde{x}^k\|^2}{\gamma_k^2}\right] + \mathbb{E}[\|\nabla f_k(x^k) - \nabla f(x^k)\|^2] + \mathbb{E}[\|\nabla f(x^k) - \nabla f(x^{k+1})\|^2]\right) \\
&\le 3\left(\bar{\alpha}\mathbb{E}\left[\frac{\|x^{k+1} - x^k\|^2}{\gamma_k^2}\right] + \sigma^2 + L^2\mathbb{E}[\|x^{k+1} - x^k\|^2]\right). \qquad (21)
\end{aligned}$$

The second inequality above holds because of the Cauchy–Schwarz inequality. In the last inequality, we used Lemma 4.7 and the assumption that f is L-Lipschitz differentiable. We want to bound $\liminf_{k\to\infty} \mathbb{E}[\frac{\|x^{k+1}-x^k\|^2}{\gamma_k^2}]$. From (20) it follows that

$$\gamma_k\left((\underline{\alpha} - 2\gamma_k L)\mathbb{E}\left[\frac{\|x^{k+1} - x^k\|^2}{4\gamma_k^2}\right] - \frac{\sigma^2}{\underline{\alpha}}\right) \le \mathbb{E}[f(x^k) - f(x^{k+1})].$$

Summing the above inequality over k yields

$$\sum_{k=0}^{\infty}\gamma_k\left((\underline{\alpha} - 2\gamma_k L)\mathbb{E}\left[\frac{\|x^{k+1} - x^k\|^2}{4\gamma_k^2}\right] - \frac{\sigma^2}{\underline{\alpha}}\right) \le f(x^0) < \infty.$$

Since $\gamma_k > 0$ and $\sum_{k=1}^{\infty} \gamma_k = \infty$, we must have

$$\liminf_{k\to\infty} (\underline{\alpha} - 2\gamma_k L)\mathbb{E}\left[\frac{\|x^{k+1} - x^k\|^2}{4\gamma_k^2}\right] - \frac{\sigma^2}{\underline{\alpha}} \le 0,$$

and thus

$$\liminf_{k\to\infty} \mathbb{E}\left[\frac{\|x^{k+1} - x^k\|^2}{\gamma_k^2}\right] \le \lim_{k\to\infty} \frac{4\sigma^2}{\underline{\alpha}(\underline{\alpha} - 2\gamma_k L)} = \frac{4\sigma^2}{\underline{\alpha}^2}.$$

Finally, from (21) it follows that

$$\liminf_{k\to\infty} \mathbb{E}[\mathrm{dist}(\mathbf{0}, \partial h(x^k))^2] \le 3\sigma^2 \left(\frac{4\bar{\alpha}}{\underline{\alpha}^2} + 1\right),$$

which completes the proof. ∎

REFERENCES

[1] D. P. Bertsekas, *Nonlinear Programming* Athena Scientific, Belmont, MA, 1999.

[2] Z. Cai, X. He, J. Sun, and N. Vasconcelos, *Deep Learning with Low Precision by Half-Wave Gaussian Quantization*, preprint, arXiv:1702.00953, 2017.

[3] E. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal rconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[4] M. Carreira-Perpinán, *Model Compression as Constrained Optimization, with Application to Neural Nets. Part* I: *General Framework*, preprint, arXiv:1707.01209, 2017.

[5] M. Carreira-Perpinán and Y. Idelbayev, *Model Compression as Constrained Optimization, with Application to Neural Nets. Part* II: *quantization*, preprint, arXiv:1707.04319, 2017.

[6] R. Chartrand and W. Yin, *Iteratively reweighted algorithms for compressive sensing*, in Proceedings of ICASSP, 2008, pp. 3869–3872.

[7] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier, *Deep Relaxation: Partial Differential Equations for Optimizing Deep Neural Networks*, preprint, arXiv:1704.04932, 2017.

[8] S. Chen, D. Donoho, and M. Saunders, *Atomic decomposition by basis pursuit*, SIAM Rev., 43 (2001), pp. 129–159.

[9] P. L. Combettes and J.-C. Pesquet, *Stochastic approximations and perturbations in forward-backward splitting for monotone operators*, Pure App. Funct. Anal., 1 (2016), pp. 13–37.

[10] M. Courbariaux, Y. Bengio, and J. David, *Binaryconnect: Training deep neural networks with binary weights during propagations*, in Proceedings of NIPS, 2015, pp. 3123–3131.

[11] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, *ImageNet: A large-scale hierarchical image database*, in Proceedings of CVPR, 2009, pp. 248–255.

[12] D. Donoho, *De-noising by soft-thresholding*, IEEE Trans. Inform. Theory, 41 (1995), pp. 613–627.

[13] J. Fan and R. Li, *Variable selection via nonconcave penalized likelihood and its oracle properties*, J. Amer. Statist. Assoc., 96 (2001), pp. 1348–1360.

[14] T. Goldstein and S. Osher, *The split Bregman method for ℓ_1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

[15] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, preprint, arXiv:1512.03385, 2015.

[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, *Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to* +1 or −1, preprint, arXiv:1602.02830, 2016.

[17] S. Ioffe and C. Szegedy, *Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, preprint, arXiv:1502.03167, 2015.

[18] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, Technical report, 2009.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton, *ImageNet classification with deep convolutional neural networks*, in Proceedings of NIPS, 2012, pp. 1097–1105.
[20] G. Lacey, G. W. Taylor, and S. Areibi, *Stochastic Layer-wise Precision in Deep Neural Networks*, preprint, arXiv:1807.00942, 2018.
[21] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.
[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, in Proc. IEEE, 86 (1998), pp. 2278–2324.
[23] C. Leng, H. Li, S. Zhu, and R. Jin, *Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM*, preprint, arXiv:1707.09870, 2017.
[24] F. Li, B. Zhang, and B. Liu, *Ternary Weight Networks*, preprint, arXiv:1605.04711, 2016.
[25] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein, *Training quantized nets: A deeper understanding*, in Proceedings of NIPS, 2017, pp. 5813–5823.
[26] Z. Li, X. Wang, X. Lv, and T. Yang, *Sep-nets: Small and Effective Pattern Networks*, preprint, arXiv:1706.03912, 2017.
[27] S. Lloyd, *Least squares quantization in PCM*, IEEE Trans. Inform. Theory, 28 (1982), pp. 129–137.
[28] B. Mordukhovich, *Variational Analysis and Generalized Differentiation* I*: Basic Theory*, Springer, New York, 2006.
[29] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bull. Soc. Math. France, 93 (1965), pp. 273–299.
[30] E. Park, J. Ahn, and S. Yoo, *Weighted-entropy-based quantization for deep neural networks*, in Proceedings of CVPR, 2017, pp. 5456–5464.
[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic Differentiation in Pytorch*, Technical report, 2017.
[32] M. Pham, P. Yin, A. Rana, S. Osher, and J. Miao, *Generalized Proximal Smoothing for Phase Retrieval*, UCLA CAM Report 18–08, 2018.
[33] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, *XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks*, preprint, arXiv:1603.05279, 2016.
[34] S. Ren, K. He, R. Girshick, and J. Sun, *Faster R-CNN: Towards real-time object detection with region proposal networks*, in Proceedings of NIPS, 2015, pp. 91–99.
[35] R. Rockafellar and R. Wets, *Variational Analysis*, Springer, New York, 2009.
[36] L. Rosasco, S. Villa, and B. C. Vũ, *Convergence of Stochastic Proximal Gradient Algorithm*, preprint, arXiv:1403.5074, 2014.
[37] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, preprint, arXiv:1409.1556, 2014.
[38] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc. Ser. B., 58 (1996), pp. 267–288.
[39] P. Yin, Y. Lou, Q. He, and J. Xin, *Minimization of ℓ_{1-2} for compressed sensing*, SIAM J. Sci. Comput., 37 (2015), pp. A536–A563.
[40] P. Yin, S. Zhang, Y. Qi, and J. Xin, *Quantization and training of low bit-width convolutional neural networks for object detection*, J. Comput. Math., to appear.
[41] W. Yin, *Analysis and generalizations of the linearized bregman method*, SIAM J. Imaging Sci., 3 (2010), pp. 856–877.
[42] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci., 1 (2010), pp. 143–168.
[43] Y. Yoshida, R. Oiwa, and T. Kawahara, *Ternary sparse xnor-net for fpga implementation*, in Proceedings of the International Symposium on Next Generation Electronics, IEEE, 2018, pp. 1–2.
[44] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, *Incremental Network Quantization: Towards Lossless CNNS with Low-Precision Weights*, preprint, arXiv:1702.03044, 2017.
[45] 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*, preprint, arXiv:1606.06160, 2016.
[46] C. Zhu, S. Han, H. Miao, and W. Dally, *Trained Ternary Quantization*, preprint, arXiv:1612.01064, 2016.