Demystifying and Generalizing BinaryConnect

Anonymous Author(s)
Affiliation
Address
email

Abstract

BinaryConnect (BC) and its many variations have become the de facto standard for neural network quantization. However, our understanding of the inner workings of BC is still quite limited. We attempt to close this gap in four different aspects: (a) we show that existing quantization algorithms, including post-training quantization, are surprisingly similar to each other; (b) we argue for proximal maps as a natural family of quantizers that is both easy to design and analyze; (c) we refine the observation that BC is a special case of dual averaging, which itself is a special case of the generalized conditional gradient algorithm; (d) consequently, we propose *ProxConnect* (PC) as a generalization of BC and we prove its convergence properties by exploiting the established connections. We conduct experiments on CIFAR-10 and ImageNet and verify that PC achieves competitive performance.

1 Introduction

Scaling up to extremely large and ever-growing datasets and models has been a main ingredient for the success of deep learning. Indeed, with the availability of big data, more computing power, convenient software, and a bag of training tricks and algorithmic innovations, the size of models that we routinely train in order to achieve state-of-the-art performance has exploded, e.g., to billions of parameters in recent language models [8]. However, high memory usage and computational cost at inference time has made it difficult to deploy these models in real-time or on resource-limited devices [e.g., 26]. The environmental impact of training and deploying these large models has also been recognized [34]. A common approach to tackle these problems is to compress a large model through quantization, i.e., replacing high-precision parameters with lower-precision ones. For example, we may constrain a subset of the weights to be binary [1, 5, 10, 27, 30, 32, 39] or ternary [24, 45]. Quantization can drastically decrease the carbon footprint of training and inference of neural networks, however, it may come at the cost of increased bias [19].

One of the main methods to obtain quantized neural networks is to encourage quantized parameters during gradient training using explicit or implicit regularization techniques, however, other methods are possible [14–16, 18, 23, 29, 43]. Besides the memory benefits, the structure of the quantization can speed up inference using, for example, faster matrix-vector products [16, 21]. Training and inference can be made even more efficient by also quantizing the activations [32] or gradients [44]. Impressive performance has been achieved with quantized networks, for example, for object detection [38] and natural language processing [37]. The theoretical underpinnings of quantized neural networks, such as when and why their performance remains reasonably well, have been actively studied [3, 12, 20, 35, 40].

BinaryConnect [BC, 10] and its many variations [e.g., 9, 32, 45] are considered the gold standard for neural network quantization. Compared to plain (stochastic) gradient descent, BC does not evaluate the gradient at the current iterate directly but rather at a quantization point using the Straight Through Estimator [6]. Despite its empirical success, BC has largely remained a "training trick" [1] and a rigorous understanding of its inner workings is yet to be found, with some preliminary steps taken

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

in Li et al. [25] for the convex setting and in Yin et al. [39] for a particular quantization set. As pointed out in Bai et al. [5], BC only evaluates gradients at the finite set of quantization points, and therefore does not exploit the rich information carried in the continuous weights network. Bai et al. [5] also observed that BC is formally equivalent to the dual averaging algorithm [28, 36], while some similarity to the mirror descent algorithm was found in Ajanthan et al. [1].

The main goal of this work is to significantly improve our understanding of BC, by connecting it with well-established theory and algorithms. In doing so we not only simplify and improve existing results but also obtain novel generalizations. We summarize our main contributions in more details:

- In §2 we show that existing gradient-based quantization algorithms are surprisingly similar to each other: the only high-level difference is at what points we evaluate the gradient and perform the update.
- In §3 we present a principled theory for constructing proximal quantizers. Our results unify previous efforts, remove tedious calculations, and bring theoretical convenience. We illustrate our theory by effortlessly designing a new quantizer that can be computed in one-pass, works for different quantization sets (binary or multi-bit), is end-to-end learnable, and includes previous attempts in [5, 10, 39] as special cases.
- In §4 we significantly extend the observation of Bai et al. [5]: BC is a nonconvex counterpart of dual averaging, and more importantly, dual averaging itself is simply the generalized conditional gradient algorithm applied to a smoothened dual. The latter fact, even in the convex case, does not appear to be widely recognized to the best of our knowledge.
- In §5, making use of the above established results, we propose *ProxConnect* (PC) as a family of algorithms that generalizes BC and we prove its convergence properties for both the convex and the nonconvex setting. We rigorously justify the diverging parameter in proximal quantizers and resolve a discrepancy between the practice and theory in [1, 5, 39].
- In §6 we verify that PC outperforms BC and ProxQuant [5] on CIFAR-10 for both fine-tuning pretrained models as well as end-to-end training. On the more challenging ImageNet dataset, PC yields competitive performance despite of no serious hyperparameter tuning.

2 Background

We consider the usual (expected) objective $\ell(\mathbf{w}) = \mathbf{E}\ell(\mathbf{w}, \boldsymbol{\xi})$, where $\boldsymbol{\xi}$ represents random sampling. For instance, $\ell(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w})$ where $\boldsymbol{\xi}$ is uniform over n training samples (or minibatches thereof), $\mathbf{w}$ is the weights of a neural network and ℓ_i may be the cross-entropy loss (of the i-th training sample). We denote a sample (sub)gradient of ℓ at $\mathbf{w}$ and $\boldsymbol{\xi}$ as $\widetilde{\nabla}\ell(\mathbf{w}) = \nabla\ell(\mathbf{w}, \boldsymbol{\xi})$ so that $\mathbf{E}\widetilde{\nabla}\ell(\mathbf{w}) = \nabla\ell(\mathbf{w})$. Throughout, we use the starred notation $\mathbf{w}^*$ for continuous weights and reserve $\mathbf{w}$ for (semi)discrete ones.

We are interested in solving the following (nonconvex) problem:

$$\min_{\mathbf{w}\in Q} \ \ell(\mathbf{w}), \tag{1}$$

where $Q \subseteq \mathbb{R}^d$ is a discrete, nonconvex quantization set. For instance, on certain low-resource devices it may be useful or even mandatory to employ binary weights, i.e., $Q = \{\pm 1\}^d$. Importantly, our goal is to compete against the non-quantized, continuous weights network (i.e. $Q = \mathbb{R}^d$). In other words, we do *not* necessarily aim to solve the (hard, combinatorial) problem (1) globally and optimally. Instead, we want to find discrete weights $\mathbf{w} \in Q$ that remain satisfactory when compared to the non-quantized continuous weights. This is how we circumvent the difficulty in (1) and more importantly how the structure of ℓ could come into aid. If ℓ is reasonably smooth, a close-by quantized weight of a locally, or even globally, optimal continuous weights will likely yield similar performance.

Tremendous progress has been made on quantizing and compressing neural networks. While it is not possible to discuss all related work, below we recall a few families of gradient-based quantization algorithms that directly motivate our work; many methods can be found in recent surveys [13, 31].

BinaryConnect (BC). Courbariaux et al. [10] were interested in binary networks where $Q = \{\pm 1\}^d$ and proposed the BinaryConnect algorithm:

$$\mathbf{w}_{t+1}^* = \mathbf{w}_t^* - \eta_t \widetilde{\nabla}\ell(\mathsf{P}(\mathbf{w}_t^*)), \tag{2}$$

where P is a projector that quantizes the continuous weights $\mathbf{w}_t^*$ either deterministically (by taking the sign) or stochastically. Note that the (sample) gradient is evaluated at the quantized weights $\mathbf{w}_t := \mathsf{P}(\mathbf{w}_t^*)$, while its continuous output, after scaled by the step size η_t, is added to the continuous weights $\mathbf{w}_t^*$. For later comparison, it is useful to break the BC update into:

$$\mathbf{w}_t = \mathsf{P}(\mathbf{w}_t^*), \qquad \mathbf{w}_{t+1}^* = \mathbf{w}_t^* - \eta_t \widetilde{\nabla} \ell(\mathbf{w}_t). \tag{3}$$

Other choices of P [e.g., 1, 39] and Q [e.g., 38, 45] have also been experimented with.

ProxQuant (PQ). Bai et al. [5] applied the usual proximal gradient to solve (1):

$$\mathbf{w}_{t+1} = \mathsf{P}(\mathbf{w}_t - \eta_t \widetilde{\nabla} \ell(\mathbf{w}_t)), \tag{4}$$

which can be similarly decomposed into:

$$\mathbf{w}_t = \mathsf{P}(\mathbf{w}_t^*), \qquad \mathbf{w}_{t+1}^* = \mathbf{w}_t - \eta_t \widetilde{\nabla} \ell(\mathbf{w}_t). \tag{5}$$

Thus, the only high-level difference between BC and PQ is that the former updates the continuous weights $\mathbf{w}_t^*$ while the latter updates the quantized weights $\mathbf{w}_t$. This seemingly minor difference turns out to cause drastically different behavior of the two. In particular, choosing P to be the Euclidean projection to Q works well for BinaryConnect but not at all for ProxQuant.

Reversing BinaryConnect. The above comparison naturally suggests a reversed variant of BC:

$$\mathbf{w}_t = \mathsf{P}(\mathbf{w}_t^*), \qquad \mathbf{w}_{t+1}^* = \mathbf{w}_t - \eta_t \widetilde{\nabla} \ell(\mathbf{w}_t^*), \tag{6}$$

which amounts to switching the continuous and quantized weights in the BC update. Similar to PQ, the choice of P is critical in this setup. To the best of our knowledge, this variant has not been formally studied before.

Post-Training Quantization. Lastly, we can also rewrite the naive post-training quantization scheme in a similar form:

$$\mathbf{w}_t = \mathsf{P}(\mathbf{w}_t^*), \qquad \mathbf{w}_{t+1}^* = \mathbf{w}_t^* - \eta_t \widetilde{\nabla} \ell(\mathbf{w}_t^*), \tag{7}$$

where we simply train the continuous network as usual and then quantize at the end. Note that the quantized weights $\mathbf{w}_t$ do not affect the update of the continuous weights $\mathbf{w}_t^*$.

3 What Makes a Good Quantizer?

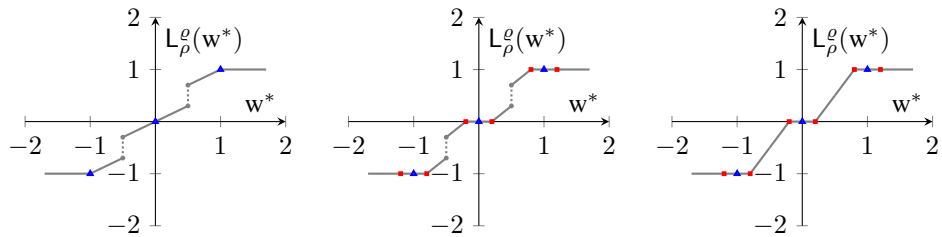
As we have seen in Section 2, the quantizer P turns out to be crucial for solving the quantization problem (1). Indeed, if $\mathsf{P} = \mathsf{P}_Q$ is the projector onto the discrete, quantization set Q, then both BC and ProxQuant only evaluate the gradient of ℓ at (the finite set of) points in Q. As a result, the methods will not be able to exploit the rich information carried in the continuous weights network, which can lead to non-convergence [5, Fig. 1b]. Since then, many semi-discrete quantizers, that turn continuous weights into more and more discrete ones, have been invented [e.g., 1, 5, 30, 39]. In this section, we present a principled way to construct quantizers that unifies previous efforts, removes tedious calculations, and also brings theoretical convenience when it comes down to analyze the convergence of the algorithms in Section 2.

Our construction is based on the proximal map $\mathsf{P}_\mathsf{r}^\nu \colon \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ of a (closed) function r:

$$\mathsf{P}_\mathsf{r}^\nu(\mathbf{w}^*) = \arg\min_{\mathbf{w}} \ \{\tfrac{1}{2\nu}\|\mathbf{w} - \mathbf{w}^*\|_2^2 + \mathsf{r}(\mathbf{w})\}, \tag{8}$$

where $\nu > 0$ is a smoothing parameter. The proximal map is well-defined as long as the function r is lower bounded by a quadratic function, in particular, when r is bounded from below. If r is proper and (closed) convex, then the minimizer on the right-hand side of (8) is uniquely attained, while for general (nonconvex) functions the proximal map may be multi-valued. If $\mathsf{r} = \iota_Q$ is the indicator function (see (15) below), then $\mathsf{P}_\mathsf{r}^\nu$ reduces to the familiar projector P_Q (with any ν). Remarkably, a complete characterization of such maps on the real line is available:

Theorem 3.1 ([41, Proposition 3])**.** *A (possibly multi-valued) map $\mathsf{P} \colon \mathbb{R} \rightrightarrows \mathbb{R}$ is a proximal map (of some function r) if and only if it is (nonempty) compact-valued, monotone and has a closed graph. The underlying function r is unique (up to addition of constants) iff P is convex-valued, while r is convex iff P is nonexpansive (i.e. 1-Lipschitz continuous).*



(a) $\rho = 0, \varrho = 0.2$. (b) $\rho = \varrho = 0.2$. (c) $\rho = 0.2, \varrho = 0$.

Figure 1: Different instantiations of the proximal map L_ρ^ϱ in (13) for $Q = \{-1, 0, 1\}$.

The sufficient and necessary conditions of Theorem 3.1 allow one to design proximal maps P directly, without needing to know the underlying function r at all (even though it is possible to integrate a version of r from P). The point is that, as far as the quantization algorithms in Section 2 are concerned, having P is enough, and hence one is excused from the tedious calculations in deriving P from r as is typical in existing works. We note in passing that the mirror map constructed in Ajanthan et al. [1, Theorem 1], such as $\tanh$, is a proximal map according to Theorem 3.1.

The next result allows us to employ *stochastic* quantizers, where in each iteration we randomly choose one of P_i to quantize the weights. Similarly, we may also apply different quantizers to different layers of a neural network.

Theorem 3.2. *Let $\mathsf{P}_i : \mathbb{R}^d \rightrightarrows \mathbb{R}^d, i = 1, \dots, k$ be proximal maps. Then, the averaged map*

$$\mathsf{P} \coloneqq \textstyle\sum_{i=1}^k \alpha_i \mathsf{P}_i, \qquad \text{where } \alpha_i \geq 0, \quad \sum_{i=1}^k \alpha_i = 1, \tag{9}$$

is also a proximal map. Similarly, the product map

$$\mathsf{P} \coloneqq \mathsf{P}_1 \times \mathsf{P}_2 \times \cdots \times \mathsf{P}_k, \quad \mathbf{w}^* = (\mathbf{w}_1^*, \dots, \mathbf{w}_k^*) \mapsto \big(\mathsf{P}_1(\mathbf{w}_1^*), \dots, \mathsf{P}_k(\mathbf{w}_k^*)\big) \tag{10}$$

is a proximal map (from $\mathbb{R}^{dk}$ to $\mathbb{R}^{dk}$).

Example 3.3. *Let Q be the quantization set (e.g. $\{\pm 1, 0\}$). Clearly, the identity map* id *and the projector P_Q are proximal maps. Therefore, their convex combination $\mathsf{P}^\nu = \frac{\mathsf{id} + \nu \mathsf{P}_Q}{1+\nu}, \nu \geq 0$, is also a proximal map, which is exactly the quantizer used in BinaryRelax [39]; see Figure 1 (a) for an illustration after the reparameterization $\varrho = \frac{\nu}{2(1+\nu)}$.*

Lastly, we mention that it is intuitively desirable to have $\mathsf{P}^\nu \to \mathsf{P}_Q$ when ν is sufficiently large, so that the quantizer eventually settles on *bona fide* discrete values in Q. This is easily achieved by letting minimizers of the underlying function r, or the (possibly larger set of) fixed points of P^ν, approach Q. We give one such construction by exploiting Theorem 3.1 and Theorem 3.2.

Let $Q_j = \{q_{j,1} \leq \cdots \leq q_{j,b_j}\} \subseteq \mathbb{R}$ be the quantization set for the j-th weight group[1] $\mathbf{w}_j^* \in \mathbb{R}^{d_j}$. Let $p_{j,k} \coloneqq \frac{q_{j,k-1} + q_{j,k}}{2}$ be the middle points. We introduce two parameters $\rho, \varrho \geq 0$ and define

$$\text{horizontal shifts} \qquad q_{j,k}^- \coloneqq p_{j,k} \vee (q_{j,k} - \rho), \;\; q_{j,k}^+ \coloneqq p_{j,k+1} \wedge (q_{j,k} + \rho), \tag{11}$$

$$\text{vertical shifts} \qquad p_{j,k}^- \coloneqq q_{j,k-1} \vee (p_{j,k} - \varrho), \;\; p_{j,k}^+ \coloneqq q_{j,k} \wedge (p_{j,k} + \varrho), \tag{12}$$

with $q_{j,1}^- = q_{j,1}$ and $q_{j,b_j}^+ = q_{j,b_j}$. Then, we define L_ρ^ϱ as the piece-wise linear map (that simply connects the points by straight lines):

$$\mathsf{L}_\rho^\varrho(w^*) = \begin{cases} q_{j,k}, & \text{if } q_{j,k}^- \leq w^* \leq q_{j,k}^+ \\ q_{j,k} + (w^* - q_{j,k}^+) \frac{p_{j,k+1}^- - q_{j,k}}{p_{j,k+1} - q_{j,k}^+}, & \text{if } q_{j,k}^+ \leq w^* < p_{j,k+1} \\ p_{j,k+1}^+ + (w^* - p_{j,k+1}) \frac{q_{j,k+1} - p_{j,k+1}^+}{q_{j,k+1}^- - p_{j,k+1}}, & \text{if } p_{j,k+1} < w^* \leq q_{j,k+1}^-, \end{cases} \tag{13}$$

for all $w^* \in \mathbf{w}_j^*$. At the middle points, $\mathsf{L}_\rho^\varrho(p_{j,k+1})$ may take any value within the two limits. Following the commonly used weight clipping in BC [10], we set $\mathsf{L}_\rho^\varrho(w^*) = q_{j,1}$ for $w^* < q_{j,1}^-$ and $\mathsf{L}_\rho^\varrho(w^*) = q_{j,b_j}$ for $w^* > q_{j,b_j}^+$. However, other choices may also work well.

[1] In this section, time subscripts are not needed. Instead, we use subscripts to indicate weight groups.

The (horizontal) parameter ρ controls the discretization vicinity within which a continuous weight will be pulled *exactly* into the discrete set Q_j, while the (vertical) parameter ϱ controls the slope (i.e. expansiveness) of each piece. It follows at once from Theorem 3.1 that L_ρ^ϱ is indeed a proximal map. In particular, setting $\rho = 0$ (hence continuous weights are only discretized in the limit) and $\varrho = \frac{\nu}{2(1+\nu)}$ we recover Example 3.3 (assuming w.l.o.g. that $q_{j,k+1} - q_{j,k} \equiv 1$). On the other hand, setting $\rho = \varrho$ leads to a generalization of the binary quantizer in Bai et al. [5], which keeps the slope to the constant 1 (while introducing jumps at middle points) and happens to be the proximal map of the distance function to Q_j [5]. Of course, setting $\rho = \varrho = 0$ yields the identity map and allows us to skip quantizing certain weights, while letting $\rho, \varrho \to \infty$ recovers the projector P_{Q_j}.

Needless to say, we may adapt the quantization set Q_j and the parameters ϱ and ρ to different weights, creating a multitude of quantization schemes. The proximal map L_ρ^ϱ of an example ternary component is visualized in Figure 1 for different choices of ρ and ϱ. Interestingly, the quantization points Q_j in L_ρ^ϱ are end-to-end learnable without incurring any significant computational overhead; see Appendix B for more details.

4 Demystifying BinaryConnect

Bai et al. [5] observed that the updates of BinaryConnect are formally the same as the dual averaging (DA) algorithm [28, 36], even though the latter algorithm was originally proposed and analyzed for convex problems. A lesser known fact is that (regularized) dual averaging itself is a special application of the generalized conditional gradient (GCG) algorithm [7, 42]. In this section, we first present the aforementioned fact, refine the observation of Bai et al. [5], and set up the stage for generalizing BinaryConnect.

4.1 Generalized Conditional Gradient is Primal-Dual

We first present the generalized conditional gradient (GCG) algorithm and point out its ability to solve simultaneously the primal and dual problems.

Let us consider the following "regularized" problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \ \{f(\mathbf{w}) \coloneqq \ell(\mathbf{w}) + \mathsf{r}(\mathbf{w})\}, \tag{14}$$

where r is a general (nonconvex) regularizer. Setting r to the indicator function of Q, i.e.

$$\mathsf{r}(\mathbf{w}) = \iota_Q(\mathbf{w}) = \begin{cases} 0, & \text{if } \mathbf{w} \in Q \\ \infty, & \text{otherwise} \end{cases}, \tag{15}$$

reduces (14) to the original problem (1). As we will see, incorporating an arbitrary r does not add any complication but will allow us to immediately generalize BinaryConnect.

Introducing the Fenchel conjugate function ℓ^* (resp. r^*) of ℓ (resp. r):

$$\ell^*(\mathbf{w}^*) \coloneqq \sup_{\mathbf{w}} \ \langle \mathbf{w}, \mathbf{w}^* \rangle - \ell(\mathbf{w}), \tag{16}$$

which is always (closed) convex even when ℓ itself is nonconvex. We derive the Fenchel–Rockafellar dual problem [e.g. 33]:

$$\min_{\mathbf{w}^*\in\mathbb{R}^d} \ \ell^*(-\mathbf{w}^*) + \mathsf{r}^*(\mathbf{w}^*), \tag{17}$$

which, unlike the original problem (14), is always a convex problem.

We apply the generalized conditional gradient algorithm [7, 42] to solving the dual problem[2] (17): Given $\mathbf{w}_t^*$, we linearize the function r^* and solve

$$\mathbf{z}_t^* = \arg\min_{\mathbf{w}^*} \ \{\ell^*(-\mathbf{w}^*) + \langle \mathbf{w}^*, \mathbf{w}_t \rangle\} = -\nabla \ell^{**}(\mathbf{w}_t), \ \ \mathbf{w}_t \coloneqq \nabla \mathsf{r}^*(\mathbf{w}_t^*), \tag{18}$$

where we used the fact that $(\nabla \ell^*)^{-1} = \nabla \ell^{**}$. Then, we take the convex combination

$$\mathbf{w}_{t+1}^* = (1-\lambda_t)\mathbf{w}_t^* + \lambda_t \mathbf{z}_t^*, \quad \text{where } \lambda_t \in [0,1]. \tag{19}$$

The following result extends Bach [4, Proposition 4.2] to any step size λ_t:

[2] GCG is usually applied to solving the primal problem (14) directly. Our choice of the dual problem here is to facilitate later comparison with dual averaging and BinaryConnect.

Theorem 4.1. *Suppose* r^* *is L-smooth (i.e.* $\nabla \mathsf{r}^*$ *is L-Lipschitz continuous), then for any* $\mathbf{w}$*:*

$$\sum_{\tau=0}^{t} \frac{\lambda_\tau}{\pi_\tau}[\ell^{**}(\mathbf{w}_\tau)+\mathsf{r}^{**}(\mathbf{w}_\tau)-\ell^{**}(\mathbf{w})-\mathsf{r}^{**}(\mathbf{w})] \leq (1-\lambda_0)\Delta(\mathbf{w},\mathbf{w}_0)+\sum_{\tau=0}^{t}\frac{\lambda_\tau^2}{2\pi_\tau}L\|\mathbf{w}_\tau^*-\mathbf{z}_\tau^*\|_2^2, \quad (20)$$

where $\mathbf{w}_t \coloneqq \nabla \mathsf{r}^*(\mathbf{w}_t^*)$*,* $\pi_t \coloneqq \prod_{\tau=1}^t(1-\lambda_\tau)$ *and* $\Delta(\mathbf{w},\mathbf{w}_t) \coloneqq \mathsf{r}^{**}(\mathbf{w}) - \mathsf{r}^{**}(\mathbf{w}_t) - \langle \mathbf{w}-\mathbf{w}_t, \mathbf{w}_t^*\rangle$.

While the convergence of $\mathbf{w}_t^*$ to the minimum of (17) is well-known (see e.g. [42]), the above result implies that a properly averaged iterate also converges to the minimum of the dual problem of (17):

Corollary 4.2. *Let* $\bar{\mathbf{w}}_t \coloneqq \sum_{\tau=0}^t \Lambda_{t,\tau}\mathbf{w}_\tau$*, where* $\Lambda_{t,\tau} \coloneqq \frac{\lambda_\tau}{\pi_\tau}/H_t$ *and* $H_t \coloneqq \sum_{\tau=0}^t \frac{\lambda_\tau}{\pi_\tau}$*. Then, abusing* $f^{**} \coloneqq \ell^{**}+\mathsf{r}^{**}$*, we have for any* $\mathbf{w}$*,*

$$\min_{0\leq\tau\leq t} f^{**}(\mathbf{w}_\tau)-f^{**}(\mathbf{w}) \leq f^{**}(\bar{\mathbf{w}}_t)-f^{**}(\mathbf{w}) \leq \frac{(1-\lambda_0)\Delta(\mathbf{w},\mathbf{w}_0)}{H_t}+\frac{L}{2}\sum_{\tau=0}^{t}\lambda_\tau\Lambda_{t,\tau}\|\mathbf{w}_\tau^*-\mathbf{z}_\tau^*\|_2^2. \quad (21)$$

Assuming $\{\mathbf{z}_\tau^*\}$ *is bounded (e.g. when* ℓ^{**} *is Lipschitz continuous), the right-hand side diminishes iff* $\lambda_t \to 0$ *and* $\sum_t \lambda_t = \infty$*.*

Thus, GCG solves problem (17) and its dual simultaneously. See Appendix A for some examples.

4.2 BC ⊆ DA ⊆ GCG

We are now ready to reveal the relationships among BinaryConnect (BC), (regularized) dual averaging (DA) and the generalized conditional gradient (GCG). Since Theorem 4.1 requires r^* to be L-smooth, in the event that it is not we resort to a smooth approximation known as the Moreau envelope:

$$\mathscr{M}_{\mathsf{r}^*}^{\mu}(\mathbf{w}^*) = \min_{\mathbf{z}^*} \ \{\tfrac{1}{2\mu}\|\mathbf{w}^*-\mathbf{z}^*\|_2^2+\mathsf{r}^*(\mathbf{z}^*)\}, \quad (22)$$

where the minimizer is exactly the proximal map $\mathsf{P}_{\mathsf{r}^*}^{\mu}$. It is well-known that $\mathscr{M}_{\mathsf{r}^*}^{\mu}$ is $(1/\mu)$-smooth and $(\mathscr{M}_{\mathsf{r}^*}^{\mu})^* = \mathsf{r}^{**}+\frac{\mu}{2}\|\cdot\|_2^2$. Thus, we apply GCG to the approximate dual problem:

$$\min_{\mathbf{w}^*\in\mathbb{R}^d} \ \ell^*(-\mathbf{w}^*)+\mathscr{M}_{\mathsf{r}^*}^{\mu}(\mathbf{w}^*), \quad (23)$$

whose own Fenchel–Rockfellar dual is:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \ \ell^{**}(\mathbf{w})+\mathsf{r}^{**}(\mathbf{w})+\tfrac{\mu}{2}\|\mathbf{w}\|_2^2. \quad (24)$$

As $\mu \to 0$, we may expect the approximate problems to "converge" to the original ones (e.g. (17)).

The updates of GCG applied to the approximate problem (23) are thus:

$$\mathbf{w}_t \coloneqq \nabla\mathscr{M}_{\mathsf{r}^*}^{\mu}(\mathbf{w}_t^*) = \frac{\mathbf{w}_t^*-\mathsf{P}_{\mathsf{r}^*}^{\mu}(\mathbf{w}_t^*)}{\mu} = \mathsf{P}_{\mathsf{r}^{**}}^{1/\mu}(\mathbf{w}_t^*/\mu) \quad (25)$$

$$\mathbf{w}_{t+1}^* = (1-\lambda_t)\mathbf{w}_t^*+\lambda_t\mathbf{z}_t^*, \quad \text{where} \quad \mathbf{z}_t^* = -\nabla\ell^{**}(\mathbf{w}_t), \quad (26)$$

which is exactly the updates of dual averaging for convex problems where $\mathsf{r}^{**} = \mathsf{r}$ and $\ell^{**} = \ell$.

Setting $\pi_t = \prod_{s=1}^t(1-\lambda_t)$ and defining $\mathsf{w}_t^* = \mathbf{w}_t^*/\pi_{t-1}$, we have (see (42) in the appendix):

$$\mathbf{w}_t = \mathsf{P}_{\mathsf{r}^{**}}^{1/\mu}(\pi_{t-1}\mathsf{w}_t^*/\mu), \qquad \mathsf{w}_{t+1}^* = \mathsf{w}_t^*-\tfrac{\lambda_t}{\pi_t}\nabla\ell^{**}(\mathbf{w}_t). \quad (27)$$

Let us now reparameterize

$$\eta_t \coloneqq \tfrac{\lambda_t}{\pi_t} \implies \lambda_t = \tfrac{\eta_t}{1+\sum_{\tau=1}^t\eta_\tau} \text{ and } \tfrac{1}{\pi_t} = 1+\textstyle\sum_{\tau=1}^t\eta_\tau. \quad (28)$$

(These relations also indicate that $\pi_t \to 0 \iff \sum_t \eta_t = \infty \iff \sum_t \lambda_t = \infty$.) If we also allow μ to change adaptively from iteration to iteration (as in dual averaging), in particular, if $\mu_t = \pi_{t-1}$, we obtain the familiar update:

$$\mathbf{w}_t = \mathsf{P}_{\mathsf{r}^{**}}^{1/\pi_{t-1}}(\mathsf{w}_t^*), \qquad \mathsf{w}_{t+1}^* = \mathsf{w}_t^*-\eta_t\nabla\ell^{**}(\mathbf{w}_t). \quad (29)$$

For nonconvex problems, we may replace r^{**} and ℓ^{**} with their nonconvex counterparts r and ℓ, respectively. We remark that r^{**} (resp. ℓ^{**}) is the largest convex function that is (pointwise) majorized by r (resp. ℓ). In particular, with $\mathsf{r} = \iota_Q$ and $\mathsf{P}_{\mathsf{r}}^{1/\mu} = \mathsf{P}_Q$ (for any μ) we recover the BinaryConnect update (3). While the parameter μ plays no role when r is an indicator function, we emphasize that in general we should use the quantizer $\mathsf{P}_{\mathsf{r}}^{1/\pi_t}$, where importantly $1/\pi_t \to \infty$ hence the quantizer converges to minimizers of r asymptotically. Neglecting this crucial detail may lead to suboptimality.

Example 4.3. *Bai et al. [5] constructed the following intriguing example:*

$$\ell(w) = \tfrac{1}{2}w^2, \;\; Q = \{\pm 1\}, \;\; \mathsf{P}_{\mathsf{r}}^{\nu}(w) = \operatorname{sign}(w)\tfrac{\epsilon|w|+\nu}{\epsilon+\nu} \text{ for } |w| \le 1\,, \tag{30}$$

and they showed non-convergence of the algorithm $w \leftarrow w - \eta\nabla\ell(\mathsf{P}_{\mathsf{r}}^{\nu}w)$, where ν is a fixed constant. If we use instead $\mathsf{P}_{\mathsf{r}}^{\nu_t}$ for some diverging ν_t, the resulting BinaryConnect, with diminishing η_t or ergodic averaging (e.g. $\bar{\mathbf{w}}_t$), would actually converge to 0 (since $\mathsf{P}_{\mathsf{r}}^{\nu_t} \to \operatorname{sign}$).

5 Proximal Connect (PC): A Generalization of BinaryConnect

We are now ready to generalize BinaryConnect by combining the results in Section 3 and Section 4. Replacing the convex envelope ℓ^{**} and r^{**} with their nonconvex counterparts, and for simplicity set $\mu_t = \pi_t = 1/\nu_t$, we obtain a *family* of algorithms which we term Proximal Connect (PC):

$$\mathbf{w}_t = \mathsf{P}_{\mathsf{r}}^{\nu_{t-1}}(\mathbf{w}_t^*), \qquad \mathbf{w}_{t+1}^* = \mathbf{w}_t^* - \eta_t\widetilde{\nabla}\ell(\mathbf{w}_t), \qquad \nu_t = 1 + \textstyle\sum_{\tau=1}^t \eta_\tau, \tag{31}$$

where the quantizer $\mathsf{P}_{\mathsf{r}}^{\nu_{t-1}}$ may be designed directly by following Section 3. We have already seen in Section 4 that BC belongs to PC by choosing $\mathsf{P}_{\mathsf{r}}^{\nu_{t-1}} = \mathsf{P}_Q$ (in which case ν_{t-1} plays no role). In Appendix A we present an even more general family and refine the convergence analysis below.

The analysis in Section 4, initially tailored for convex functions, immediately generalizes to the nonconvex algorithm PC (for both nonconvex ℓ and nonconvex r and for stochastic gradient $\widetilde{\nabla}\ell$):

Theorem 5.1. *Fix any* $\mathbf{w}$*, the iterates in* (31) *satisfy:*

$$\sum_{\tau=s}^{t} \eta_\tau[\langle \mathbf{w}_\tau - \mathbf{w}, \widetilde{\nabla}\ell(\mathbf{w}_\tau)\rangle + \mathsf{r}(\mathbf{w}_\tau) - \mathsf{r}(\mathbf{w})] \le \tfrac{\Lambda_s^t}{2}\|\mathbf{w}\|_2^2 + \Delta_{s-1}(\mathbf{w}) - \Delta_t(\mathbf{w}) + \sum_{\tau=s}^{t}\Delta_\tau(\mathbf{w}_\tau), \tag{32}$$

where $\Lambda_s^t \coloneqq \sum_{\tau=s}^t \eta_\tau/\nu_\tau$ and $\Delta_\tau(\mathbf{w}) \coloneqq \mathsf{r}_\tau(\mathbf{w}) - \mathsf{r}_\tau(\mathbf{w}_{\tau+1}) - \langle \mathbf{w} - \mathbf{w}_{\tau+1}, \mathbf{w}_{\tau+1}^*\rangle$ is the Bregman divergence induced by the (possibly nonconvex) function $\mathsf{r}_\tau(\mathbf{w}) \coloneqq \nu_\tau \cdot \mathsf{r}(\mathbf{w}) + \frac{1}{2}\|\mathbf{w}\|_2^2$.

The summand on the left-hand side of (32) is related to the duality gap in Yu et al. [42], which is a natural measure of stationarity for the nonconvex problem (14). Indeed, it reduces to the familiar ones when convexity is present:

Corollary 5.2. *For convex ℓ and any* $\mathbf{w}$*, the iterates in* (31) *satisfy:*

$$\min_{\tau=s,\dots,t} \mathbf{E}[f(\mathbf{w}_\tau) - f(\mathbf{w})] \le \tfrac{1}{\sum_{\tau=s}^t \eta_\tau} \cdot \mathbf{E}\big[\tfrac{\Lambda_s^t}{2}\|\mathbf{w}\|_2^2 + \Delta_{s-1}(\mathbf{w}) - \Delta_t(\mathbf{w}) + \textstyle\sum_{\tau=s}^t \Delta_\tau(\mathbf{w}_\tau)\big]. \tag{33}$$

If r is also convex, then

$$\min_{\tau=s,\dots,t} \mathbf{E}[f(\mathbf{w}_\tau) - f(\mathbf{w})] \le \tfrac{1}{\sum_{\tau=s}^t \eta_\tau} \cdot \mathbf{E}\big[\tfrac{\Lambda_s^t}{2}\|\mathbf{w}\|_2^2 + \Delta_{s-1}(\mathbf{w}) + \textstyle\sum_{\tau=s}^t \tfrac{\eta_\tau^2}{2}\|\widetilde{\nabla}\ell(\mathbf{w}_\tau)\|_2^2\big], \tag{34}$$

where the right-hand side diminishes iff $\eta_t \to 0$ and $\sum_t \eta_t = \infty$ (assuming boundedness of the stochastic gradient), in particular if $\eta_t = O(1/\sqrt{t})$ and say $s = t/2$.

Our bound (33) improves Li et al. [25, Thm 3], who only dealt with $\mathsf{r} = \iota_Q$, see Appendix A for details. From (32) we note some trade-off in choosing the step size η_τ: a larger η_τ may force the summand of the left-hand side to diminish faster but on the other hand loosens the right-hand side, since Δ_τ depends on $\mathbf{w}_{\tau+1}^*$, whose magnitude may accumulate proportionally to η_τ (see (31)). A detailed analysis may need to take specific properties of r or P into account [39].

It is worthwhile to point out one important difference between ProxQuant and ProxConnect: Bai et al. [5] proved convergence (to some notion of stationarity) of ProxQuant for a fixed quantizer, i.e. $\mathsf{P}_{\mathsf{r}}^{\nu}$ for a fixed ν, but their experiments relied on incrementing ν so that their quantizer approaches the projector P_Q. This creates some discrepancy between theory and practice. The same comment also applies to [1]. In contrast, ProxConnect is derived from a rigorous theory that automatically justifies a diverging ν, see (31). In particular, choosing a constant step size $\eta_\tau \equiv \eta_0$ would lead to $\nu_t \propto t$, matching the current practice that is rigorously justifiable if r is strongly convex (see Appendix A). For an illustrative comparison of BinaryConnect, ProxQuant, and ProxConnect, see Figure 2.

Connection to Existing Algorithms. Besides the obvious BC, we can recover many other quantization algorithms by setting P in a specific way. As it turns out, many of these algorithms can actually be recovered with our proposed proximal quantizer L_ρ^ϱ from Section 3; see Table 1 for a summary.

reverseProxConnect. In Section 2 we discussed the idea of reversing BinaryConnect. As PC generalizes BinaryConnect, we consider reverseProxConnect (rPC) as a generalization instead.

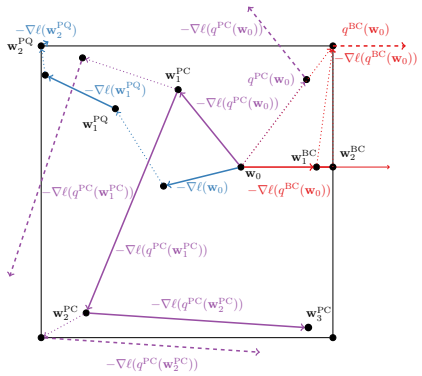


Figure 2: Three iterations of BinaryConnect (red), ProxQuant (blue) and ProxConnect (purple). Solid, dashed, and dotted lines visualize gradient updates, straight-through gradients, and quantization maps, respectively. BC gets stuck (after clipping) as it only ever evaluates the gradient of the top-right quantization point which is perpendicular to the "quantization line". PQ evaluates gradients directly at the current iterate whereas BC and PC evaluate gradients at the output of their quantization maps and send them "straight-through". The second pre-quantization iterate of PQ is close to the top-left quantization point, and therefore gets pushed to it. Because the gradient at this point is small, PQ converges. PC, on the other hand, does not directly get pushed to quantization points but rather just evaluates their gradients when close. This feature might help to identify quantization points that have a sudden increase in function value.

Table 1: A sample summary of existing quantization algorithms. The PC column indicates if ProxConnect generalized the method. The ϟ- and 🎓-columns respectively indicate if the quantization map can be computed in one-pass and if it has learnable parameters. The L_ρ^ϱ-column indicates whether or not the method can be realized using our proposed proximal map; a 💎 indicates that we propose to use L_ρ^ϱ (with Q) which would result in a similar algorithm as the original one; a 🖇 indicates a smoothed version of L_ρ^ϱ. †: Ternary quantization methods generally use a "cutoff" that is not exactly in between the quantization points to improve performance.

Method	PC	ϟ	L_ρ^ϱ	ρ, ϱ	🎓	Q
ProxQuant-Binary-L_1 [5]	✗	✓	✓	↗	✗	$\{-1, 1\}$
ProxQuant-Ternary [5]	✗	✗	💎	↗	✗	$\{-1, 0, 1\}$
BinaryConnect [10]	✓	✓	✓	∞	✗	$\{-1, 1\}$
BinaryRelax [39]	✓	✓	✓	0, ↗	✗	$\{-1, 1\}$
BinaryWeight [32]	✓	✓	💎	∞	✓	$\{q_{j,1}, q_{j,2}\}$
MirrorDescentView [1]	✓	✓	🖇	-	✗	$\{-1, 1\}$
TrainedTernary† [45]	✓	✓	✓	∞	✓	$\{q_{j,1}, 0, q_{j,2}\}$
TernaryWeight† [24]	✓	✓	💎	∞	✓	$\{-q_{j,1}, 0, q_{j,1}\}$
ProxConnect (ours)	-	✓	✓	↗	✓	Q_j

6 Experiments

6.1 Image Classification on CIFAR-10

We perform image classification on CIFAR-10 [22] using ResNet20 and ResNet56 [17], comparing BinaryConnect [10] with ProxQuant [5] and (reverse)ProxConnect using our proposed proximal operator L_ρ^ϱ. For fair comparison, we set $\rho = \varrho$ as this resembles the original ProxQuant algorithm. Similar to Bai et al. [5], we increase the parameter ρ linearly: $\rho_t = (1 + t/B)\,\rho_0$. In contrast to Bai et al. [5], however, we increase ρ after every gradient step rather than after every epoch as is more in line with our analysis. We treat ρ_0 as a hyperparameter for which we conduct a small grid search. We consider binary, ternary and quaternary ($Q = \{-1, -0.3, 0.3, 1\}$) quantization. Details for all CIFAR-10 experiments can be found in Appendix C.

Fine-Tuning Pretrained Models. In this experiment, all quantization algorithms are initialized with a pretrained ResNet. The test accuracies of the pretrained ResNet20 and ResNet56 are 92.01 and 93.01, respectively. Table 2 shows the final test accuracy for the different models: for ProxQuant and (reverse)ProxConnect we respectively picked the best ρ_0 values; results for all ρ_0 values can be found in Appendix C. (Reverse)ProxConnect outperforms the other two methods on all six settings.

End-To-End Training. To save computational costs, it is important that quantization algorithms also perform well when they are not initialized with pretrained full-precision model. We therefore

Table 2: Fine-tuning pretrained ResNets. Final test accuracy: mean and std over three runs.

Model	Quantization	BC [10]	PQ [5]	rPC (ours)	PC (ours)
ResNet20	Binary	**90.31** (0.00)	89.94 (0.10)	89.98 (0.17)	**90.31** (0.21)
	Ternary	74.95 (0.16)	91.46 (0.06)	**91.47** (0.19)	91.37 (0.18)
	Quaternary	91.43 (0.07)	91.43 (0.21)	91.43 (0.06)	**91.81** (0.14)
ResNet56	Binary	92.22 (0.12)	92.33 (0.06)	92.47 (0.29)	**92.65** (0.16)
	Ternary	74.68 (1.4)	93.07 (0.02)	92.84 (0.11)	**93.25** (0.12)
	Quaternary	93.20 (0.06)	92.82 (0.16)	92.91 (0.26)	**93.42** (0.12)

Table 3: End-to-end training of ResNets. Final test accuracy: mean and std over three runs.

Model	Quantization	BC [10]	PQ [5]	rPC (ours)	PC (ours)
ResNet20	Binary	87.51 (0.21)	81.59 (0.75)	81.82(0.32)	**89.92** (0.65)
	Ternary	27.10 (0.21)	47.98 (1.30)	47.17(1.94)	**84.09** (0.16)
	Quaternary	89.91 (0.09)	85.29 (0.09)	85.05(0.27)	**90.17** (0.14)
ResNet56	Binary	89.79 (0.45)	86.13 (1.71)	86.25 (1.50)	**91.26** (0.59)
	Ternary	30.31 (7.79)	50.54 (3.68)	42.95 (1.57)	**84.36** (0.75)
	Quaternary	90.69 (0.57)	87.81 (1.60)	87.30 (1.02)	**91.70** (0.14)

compare the four methods for randomly initialized models; see Table 3 for the results. ProxConnect outperforms all other methods on all six tasks. Interestingly, ProxQuant and reverseProxConnect perform considerably worse for all six tasks when compared to fine-tuning. The performance drop of BinaryConnect and ProxConnect when compared to fine-tuning is only significant for ternary quantization. We found that ProxQuant and reverseProxConnect are very sensitive to the choice of ρ_0, whereas ProxConnect is stable in this regard; see Appendix C.

6.2 Image Classification on ImageNet

We perform a small study on ImageNet [11] using ResNet18 [17]. To save computational resources and because PC performed better than rPC on CIFAR-10, we only compare PC with BC on binary quantization. As can be seen in Table 4, BC performs slightly better for fine-tuning whereas PC performs slightly better for end-to-end training. This is not an exhaustive study, but rather a first indication that PC can yield competitive performance on large scale datasets. For more details on the experiments, see Appendix D.

Table 4: Fine-tuning (left) and end-to-end training (right). Final test accuracy: mean and standard deviation over three runs.

BC [10]	PC (ours)		BC [10]	PC (ours)	
	$\rho_0 = 2\text{e}{-2}$	$\rho_0 = 4\text{e}{-2}$		$\rho_0 = 2.5\text{e}{-3}$	$\rho_0 = 5\text{e}{-3}$
65.84 (0.04)	65.44 (0.13)	65.70 (0.04)	63.79 (0.12)	**63.89** (0.14)	63.67 (0.12)

7 Conclusion

Capitalizing on a principled approach for designing quantizers and a surprising connection between BinaryConnect and the generalized conditional gradient (GCG) algorithm, we proposed ProxConnect as a unification and generalization of existing neural network quantization algorithms. Our analysis refined prior convergence guarantees and our experiments confirmed the competitiveness of ProxConnect. In the future we plan to apply ProxConnect to training other models, for example, transformers and generative models. The connection with GCG also opens the possibility for further acceleration.

References

[1] Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richard Hartley, and Puneet Dokania. Mirror Descent View for Neural Network Quantization. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[2] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal. An Empirical study of Binary Neural Networks' Optimisation. In *International Conference on Learning Representations*, 2019.

[3] Alexander G. Anderson and Cory P. Berg. The High-Dimensional Geometry of Binary Neural Networks. In *International Conference on Learning Representations*, 2018.

[4] Francis Bach. Duality Between Subgradient and Conditional Gradient Methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.

[5] Yu Bai, Yu-Xiang Wang, and Edo Liberty. ProxQuant: Quantized Neural Networks via Proximal Operators. In *International Conference on Learning Representations*, 2018.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432*, 2013.

[7] Kristian Bredies and Dirk A. Lorenz. Iterated Hard Shrinkage for Minimization Problems with Sparsity Constraints. *SIAM Journal on Scientific Computing*, 30(2):657–683, 2008.

[8] Tom Brown and et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020.

[9] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. MetaQuant: Learning to Quantize by Learning to Penetrate Non-differentiable Quantization. In *Advances in Neural Information Processing Systems*, 2019.

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, volume 28, pages 3123–3131, 2015.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[12] Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. On the Universal Approximability and Complexity Bounds of Quantized ReLU Neural Networks. In *International Conference on Learning Representations*, 2019.

[13] Yunhui Guo. A Survey on Methods and Theories of Quantized Neural Networks. *arXiv:1808.04752*, 2018.

[14] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *International Conference on Machine Learning*, 2015.

[15] Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and Chang Xu. Training Binary Neural Networks through Learning with Noisy Supervision. In *International Conference on Machine Learning*, 2020.

[16] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[18] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization. In *Advances in Neural Information Processing Systems*, 2019.

[19] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. Characterising Bias in Compressed Models. *arXiv:2010.03058*, 2020.

[20] Lu Hou, Ruiliang Zhang, and James T. Kwok. Analysis of Quantized Models. In *International Conference on Learning Representations*, 2019.

[21] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[22] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images, 2009. The MIT License.

[23] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely Low Bit Neural Network: Squeeze the Last Bit Out With ADMM. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Fengfu Li, Bo Zhang, and Bin Liu. Ternary Weight Networks. *arXiv:1605.04711*, 2016.

[25] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training Quantized Nets: A Deeper Understanding. In *Advances in Neural Information Processing Systems*, 2017.

[26] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, and Song Han. MCUNet: Tiny Deep Learning on IoT Devices. *Advances in Neural Information Processing Systems*, 2020.

[27] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2020.

[28] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[29] E. Park, J. Ahn, and S. Yoo. Weighted-Entropy-Based Quantization for Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7197–7205, 2017.

[30] Vahid Partovi Nia and Mouloud Belbahri. Binary Quantizer. *Journal of Computational Vision and Imaging Systems*, 4(1):3, 2018.

[31] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary Neural Networks: A Survey. *Pattern Recognition*, 105:107281, 2020.

[32] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, 2016.

[33] R. Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*. Springer, 1998.

[34] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[35] Yanzhi Wang, Zheng Zhan, Liang Zhao, Jian Tang, Siyue Wang, Jiayu Li, Bo Yuan, Wujie Wen, and Xue Lin. Universal Approximation Property and Equivalence of Stochastic Computing-Based Neural Networks and Binary Neural Networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[36] Lin Xiao. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[37] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating Multi-bit Quantization for Recurrent Neural Networks. In *International Conference on Learning Representations*, 2018.

[38] P Yin, S Zhang, YY Qi, and J Xin. Quantization and Training of Low Bit-Width Convolutional Neural Networks for Object Detection. *Journal of Computational Mathematics*, 37(3), 2019.

[39] Penghang Yin, Shuai Zhang, Jiancheng Lyu, Stanley Osher, Yingyong Qi, and Jack Xin. BinaryRelax: A Relaxation Approach for Training Deep Neural Networks with Quantized Weights. *SIAM Journal on Imaging Sciences*, 11(4):2205–2223, 2018.

[40] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. In *International Conference on Learning Representations*, 2019.

[41] Yaoliang Yu, Xun Zheng, Micol Marchetti-Bowick, and Eric Xing. Minimizing Nonconvex Non-Separable Functions. In *Artificial Intelligence and Statistics*, 2015.

[42] Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Generalized Conditional Gradient for Structured Sparse Estimation. *Journal of Machine Learning Research*, 18:1–46, 2017.

[43] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental Network Quantization: Towards Lossless CNNs with Low-precision Weights. In *International Conference on Learning Representations*, 2017.

[44] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv:1606.06160*, 2016.

[45] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained Ternary Quantization. In *International Conference on Learning Representations*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] As discussed, the goal of our paper is to improve the understanding of BinaryConenct [10] and obtain novel generalizations. We therefore focus more on theory than empirical results. As a result, we only did extensive experiments on one dataset (CIFAR-10) with one architecture (ResNet20/56). As discussed in Section 6.2, our experiments on ImageNet are not exhaustive; both due to environmental reasons as well as limited availability of compute. In the future, we aim to do more exhaustive testing of ProxConnect (different architectures and problems), while keeping in mind the carbon footprint that comes with it.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the first paragraph of Section 1: quantization leads to decreased carbon footprint for training and inference (positive societal impact), however, it may come at the cost of increased bias (negative societal impact).
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] All proofs are provided in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will provide the code upon publication. We only ran experiments on publicly available datasets.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6, Appendix C and Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments are over three random seeds; we provide mean and standard deviation for all results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The amount of compute in the exploration stage of our project was insignificant compared to the experiments reported in this work. See Appendix C.6 and Appendix D.6 for the total amount of compute we used for the experiments reported in this work.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 6.1 and Section 6.2.
   (b) Did you mention the license of the assets? [Yes] Yes, the license of CIFAR-10 (MIT license) is included in the bibliography entry. The ImageNet dataset is "is available for free to researchers for non-commercial use".
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We will provide the code upon publication.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]