ProxQuant: Quantized Neural Networks via Proximal Operators

Yu Bai Yu-Xiang Wang Edo Liberty

September 21, 2018

Abstract

Deep neural networks are often desired in environments with limited memory and computational power (such as mobile devices), where it is beneficial to perform model quantization – training networks with low-precision weights. A key mechanism commonly used in training quantized nets is the straight-through gradient method, which enables back-propagation through the quantization mapping. Despite its success, little is understood why the straight-through gradient method works, especially in low-bit scenarios such as training binary networks.

We propose an alternative approach, ProxQuant, that formulates quantized network training as a regularized learning problem instead and optimizes via the prox-gradient method. ProxQuant does back-propagation on the underlying full-precision vector and applies an efficient prox operator in between stochastic gradient steps to encourage *quantizedness*. For quantizing ResNets and LSTMs, ProxQuant outperforms state-of-the-art results on binary quantization and is on par with state-of-the-art on multi-bit quantization. We further show that ProxQuant suffers from less optimization instability in the binary case through a sign change experiment. Our results challenge the indispensability of the straight-through gradient method and demonstrate that ProxQuant is a powerful alternative.

1 Introduction

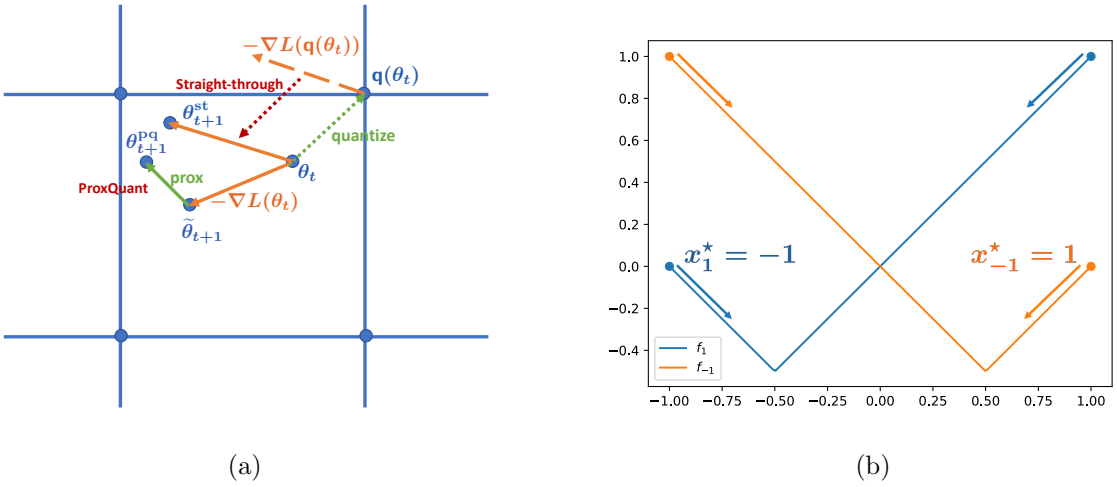
Deep neural networks (DNNs) have achieved impressive results in various machine learning tasks [7]. High-performance DNNs typically have over tens of layers and millions of parameters, resulting in a high memory usage and a high computational cost at inference time. However, these networks are often desired in environments with limited memory and computational power (such as mobile devices), in which case we would like to compress the network into a smaller, faster network with comparable performance.

A popular way of achieving such compression is through quantization – training networks with low-precision weights and/or activation functions. In a quantized neural network, each weight and/or activation can be representable in k bits, with a possible codebook of negligible additional size compared to the network itself. For example, in a binary neural network ($k = 1$), the weights are restricted to be in $\{\pm 1\}$. Compared with a 32-bit single precision float, a quantized net reduces the memory usage to $k/32$ of a full-precision net with the same architecture [8, 5, 17, 12, 23, 24]. In addition, the structuredness of the quantized weight matrix can often enable faster matrix-vector product, thereby also accelerating inference [12, 9].

Typically, training a quantized network involves (1) the design of a *quantizer* q that maps a full-precision parameter to a k-bit quantized parameter, and (2) the *straight-through gradient method* [5] that enables back-propagation from the quantized parameter back onto the original full-precision parameter, which is critical to the success of quantized network training. With quantizer q, an iterate of the straight-through gradient method (see Figure 1a) proceeds as $\theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\mathsf{q}(\theta_t))$,

and $\mathsf{q}(\widehat{\theta})$ (for the converged $\widehat{\theta}$) is taken as the output model. For training binary networks, choosing $\mathsf{q}(\cdot) = \mathrm{sign}(\cdot)$ gives the BinaryConnect method [5].

Though appealingly simple and empirically effective, it is information-theoretically rather mysterious why the straight-through gradient method works well, at least in the binary case: while the goal is to find a parameter $\theta \in \{\pm 1\}^d$ with low loss, the algorithm only has access to stochastic gradients at $\{\pm 1\}^d$. As this is a discrete set, *a priori*, gradients in this set do not contain much information about the function values. Indeed, a simple one-dimensional example (Figure 1b) shows that BinaryConnect fails to find the minimizer of fairly simple convex Lipschitz functions in $\{\pm 1\}$, due to a lack of gradient information in between.



(a) (b)

Figure 1: (a) Comparison of the straight-through gradient method and our ProxQuant method. The straight-through method computes the gradient at the quantized vector and performs the update at the original real vector; ProxQuant performs a gradient update at the current real vector followed by a prox step which encourages quantizedness. (b) A two-function toy failure case for BinaryConnect. The two functions are $f_1(x) = |x + 0.5| - 0.5$ (blue) and $f_{-1}(x) = |x - 0.5| - 0.5$ (orange). The derivatives of f_1 and f_{-1} coincide at $\{-1, 1\}$, so any algorithm that only uses this information will have identical behaviors on these two functions. However, the minimizers in $\{\pm 1\}$ are $x^\star_1 = -1$ and $x^\star_{-1} = 1$, so the algorithm must fail on one of them.

Our proposal is motivated by the following observations: (1) the straight-through gradient method is exactly a projected SGD with *lazy projection* onto $\{\pm 1\}^d$ (Section 2.2), and (2) this projection can be viewed as a proximal operator with respect to a suitable quantization-encouraging regularizer and infinite regularization strength (Section 2.3). Hence the straight-through gradient method is a special case, among many, of the prox-gradient method on a suitable regularized learning problem for quantization, and there is no prior reason that this is the best version for any given problem.

Our contribution can be summarized as follows.

- We formulate the problem of training quantized networks as a regularized learning problem, and propose to optimize via the prox-gradient method (Section 3). Our method, ProxQuant, adds a simple prox step with respect to a quantization-inducing regularizer after each stochastic gradient step (Figure 1a). Our framework has the straight-through gradient method as a special case with lazy-projection and $\lambda = \infty$, but we advocate and demonstrate the advantage of a non-lazy projection and a finite λ.

- We present a unified framework through the Wasserstein distance for deriving regularizers for binary, ternary, and multi-bit quantization (Section 3.1). In each case, the prox-operator (either exact or approximate) can be efficiently implemented. We propose a homotopy method for choosing an increasing regularization strength to enforce exact quantizedness (Section 3.2).

- We demonstrate the effectiveness and flexibility of ProxQuant through systematic experiments on (1) image classification with ResNets (Section 4.1); (2) language modeling with LSTM (Section 4.2). The ProxQuant method outperforms the state-of-the-art results for binary quantization and is comparable with the state-of-the-art for ternary and multi-bit quantization.

- For binary nets, we show the advantage of ProxQuant over BinaryConnect through the sign change metric (Section 5). Compared with BinaryConnect, ProxQuant finds higher-performance binary nets that is also closer to the initialization. This suggests that the optimization stability of ProxQuant is better than BinaryConnect (and in general the straight-through gradient method).

1.1 Prior work

Methodologies Han et al. [8] propose Deep Compression, which compresses a DNN via sparsification, nearest-neighbor clustering, and Huffman coding. This architecture is then made into a specially designed hardware for efficient inference [9]. In a parallel line of work, Courbariaux et al. [5] proposes BinaryConnect that enables the training of binary neural networks, which is then extended into ternary [14, 24]. Training and inference on quantized nets can be made more efficient by also quantizing the activation [12, 17, 23], and such networks have achieved impressive performance on large-scale tasks such as ImageNet classification [17]. In the NLP land, quantized language models have been successfully trained using alternating multi-bit quantization [22].

Theories Li et al. [15] prove the convergence rate of stochastic rounding and BinaryConnect on convex problems and demonstrate the advantage of BinaryConnect over stochastic rounding on non-convex problems. Anderson and Berg [1] demonstrate the effectiveness of binary networks through the observation that the angles between high-dimensional vectors are approximately preserved when binarized, and thus high-quality feature extraction with binary weights is possible. Ding et al. [6] show a universal approximation theorem for quantized ReLU networks.

Principled methods Sun and Sun [18] implements the Wasserstein regularization term $W(\mathcal{L}(\theta), P_0)$ via the adversarial representation, similar as in Wasserstein GANs [2]. This method can handle the case when the Wasserstein distance does not have a closed-form expression, but might be hard to tune due to the instability of the inner maximization problem.

While we are preparing this manuscript, we discovered the independent work of Carreira-Perpinán [3], Carreira-Perpinán and Idelbayev [4]. They formulate quantized network training as a constrained optimization problem and propose to solve them via augmented Lagrangian methods. Although a special case of our method (W_2^2 distance + approximate prox) is equivalent to the first-order variant of their method (one gradient step per inner minimization), our Wasserstein framework offers more options such as the non-smooth regualrizer for binary nets. Further, from an optimization perspective, our views are largely complementary: they treat the quantization as a constraint, whereas we encourage quantization through a regularizer. Due to time constraints, we did not do experimental comparison (they only reported results on VGG whereas we focus on

ResNets) – as they solve a full augmented Lagrangian minimization in between each compression step, successful training of their LC algorithm will at least require a careful tuning of this inner optimization procedure.

1.2 Notation

Throughout the paper, we let $\theta \in \mathbb{R}^d$ denote the parameters of a neural network, $L(\theta)$ denote the loss function over the entire dataset (the empirical risk), and $\widetilde{\nabla} L$ denote the stochastic gradient of L (e.g. over a minibatch). For the regularization method, we denote the set of quantized parameters by $\mathcal{Q}$, the regularizer by $R(\theta)$ and the regularization strength by λ. The learning rates are denoted by η_t for $t \geq 0$. We let $\mathrm{Proj}_S : \mathbb{R}^d \to \mathbb{R}^d$ denote the standard Euclidean projection onto a set $S \subset \mathbb{R}^d$ and prox_f denote the proximal operator w.r.t. function f (details in Section 2.3). The p-Wasserstein distance is denoted as W_p. We will restrict attention to Wasserstein distances on $\mathbb{R}$ and $p \in \{1, 2\}$.

2 Preliminaries

A central problem with training quantized models is that they involve a discrete parameter space and hence efficient local-search methods are often prohibitive. For example, the problem of training a binary neural network is to minimize $L(\theta)$ for $\theta \in \{\pm 1\}^d$. Projected SGD on this set will not move unless with an unreasonably large stepsize [15], whereas greedy nearest-neighbor search requires d forward passes which is intractable for neural networks where d is on the order of millions. Alternatively, quantized training can also be cast as minimizing $L(\mathsf{q}(\theta))$ for $\theta \in \mathbb{R}^d$ and an appropriate *quantizer* q that maps a real vector to a nearby quantized vector, but $\theta \mapsto \mathsf{q}(\theta)$ is often non-differentiable and piecewise constant (such as the binary case $\mathsf{q}(\cdot) = \mathrm{sign}(\cdot)$), and thus back-propagation through q does not work.

2.1 Straight-through gradient estimates

The pioneering work of BinaryConnect [5] proposes to solve this problem via the *straight-through gradient estimate*, that is, propagate the gradient with respect to $\mathsf{q}(\theta)$ unaltered to θ, i.e. to let $\frac{\partial L}{\partial \theta} := \frac{\partial L}{\partial \mathsf{q}(\theta)}$. One iterate of the straight-through gradient method (with the SGD optimizer) is

$$\theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\mathsf{q}(\theta_t)).$$

This enables the real vector θ to move in the entire Euclidean space, and taking $\mathsf{q}(\theta)$ at the end of training gives a valid quantized model. Such a customized back-propagation rule yields good empirical performance in training quantized nets and have thus become a standard practice [5, 24, 22].

While it makes sense (by gradient Lipschitzness) that the gradients at θ and $\mathsf{q}(\theta)$ should be close when $\mathsf{q}(\theta)$ and θ themselves are close, this is often not the case – in training binary nets we have $\theta \in [-1, 1]^d$ and $\mathsf{q}(\theta) = \mathrm{sign}(\theta)$, and $\|\theta - \mathsf{q}(\theta)\|_2$ can be large in general. In this case, the Lipschitz gradient explanation ceases to hold, and it is quite mysterious why BinaryConnect works well in finding a good quantized net at convergence.

2.2 Straight-through gradient as lazy projection

Our first observation is that the straight-through gradient estimate is equivalent to a *dual-averaging* method, or a projected SGD with *lazy projection* [21]. In the binary case, we wish to minimize

$L(\theta)$ over $\mathcal{Q} = \{\pm 1\}^d$, and the dual-averaging method proceeds as

$$\begin{cases} \widetilde{\theta}_t = \mathrm{Proj}_{\mathcal{Q}}(\theta_t) = \mathrm{sign}(\theta_t) = \mathsf{q}(\theta_t), \\ \theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\widetilde{\theta}_t). \end{cases} \tag{1}$$

Written compactly, this is $\theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\mathsf{q}(\theta_t))$, which is exactly the straight-through gradient estimate: take the gradient at the quantized vector and perform the update on the original real vector. Although convergence guarantee for dual-averaging methods are not known for non-convex L and $\mathcal{Q}$, its nice form can serve as an explanation why the straight-through gradient estimate works well in practice.

2.3 Projection as proximal operator of a regularizer

We take a broader point of view that a projection is also a proximal operator with a suitable regularizer, to allow more generality and to motivate our proposed algorithm. Given any set $\mathcal{Q}$, one could identify a regularizer $R : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ such that the following hold:

$$R(\theta) = 0, \quad \forall \theta \in \mathcal{Q} \quad \text{and} \quad R(\theta) > 0, \quad \forall \theta \notin \mathcal{Q}. \tag{2}$$

In the case $\mathcal{Q} = \{\pm 1\}^d$ for example, one could take

$$R(\theta) = R_{\mathrm{bin}}(\theta) = \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\}. \tag{3}$$

The proximal operator (or prox operator) with respect to R and strength $\lambda > 0$ is

$$\mathrm{prox}_{\lambda R}(\theta_0) := \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \left\{ \frac{1}{2} \|\theta - \theta_0\|_2^2 + \lambda R(\theta) \right\}.$$

When $\lambda = \infty$, the argmin has to satisfy $R(\theta) = 0$, i.e. $\theta \in \mathcal{Q}$, and the prox operator is to minimize $\|\theta - \theta_0\|_2^2$ over $\theta \in \mathcal{Q}$, which is the Euclidean projection onto $\mathcal{Q}$. Hence, projection is also a prox operator with $\lambda = \infty$, and the straight-through gradient estimate is equivalent to a lazy proximal gradient descent with and $\lambda = \infty$.

While the prox operator with $\lambda = \infty$ correponds to "hard" projection onto the discrete set $\mathcal{Q}$, when $\lambda < \infty$ it becomes a "soft" projection that moves towards $\mathcal{Q}$. Compared with the hard projection, the prox operator is less aggressive and has the potential advantage of avoiding overshoot early in training. Further, as the prox operator does not strictly enforce quantizedness, it is in principle able to query the gradients at every point in the space, and therefore has access to more information than the straight-through gradient method.

3 Quantized net training via regularized learning

We propose the ProxQuant algorithm, which adds a quantization enforcing regularization onto the loss and optimizes via the prox-gradient method with non-lazy projection and a finite λ. The prototypical version of ProxQuant is described in Algorithm 1.

Algorithm 1 ProxQuant: Prox-gradient method for quantized net training

Require: Regularizer R that enforces desired quantizedness, initialization θ_0, learning rates $\{\eta_t\}_{t\geq 0}$, regularization strengths $\{\lambda_t\}_{t\geq 0}$

while not converged **do**

Perform the prox-gradient step

$$\theta_{t+1} = \arg\min_{\theta\in\mathbb{R}^d} \left\{ L(\theta_t) + \left\langle \theta - \theta_t, \widetilde{\nabla} L(\theta_t) \right\rangle + \frac{1}{2\eta_t} \|\theta - \theta_t\|_2^2 + \lambda_t R(\theta) \right\} \tag{4}$$

$$= \mathrm{prox}_{\eta_t \lambda_t R} \left(\theta_t - \eta_t \widetilde{\nabla} L(\theta_t) \right). \tag{5}$$

The inner SGD step in (5) can be replaced by any preferred stochastic optimization method such as Momentum SGD or Adam [13].

end while

Compared to usual full-precision training, ProxQuant only adds a prox step after each stochastic gradient step, hence can be implemented straightforwardly upon existing full-precision training. As the prox step does not need to know how the gradient step is performed, our method adapts to other stochastic optimizers as well such as Adam. Further, each iteration is a prox-gradient step over the objective $L(\theta) + \lambda_t R(\theta)$ with step-size η_t, and by choosing (η_t, λ_t) we obtain a joint control over the speed of training and falling onto the quantized set.

In the remainder of this section, we describe a flexible class of quantization-inducing regularizers and a homotopy approach for choosing the regularization strengths $\{\lambda_t\}$.

3.1 Wasserstein regularization

An important problem in practice is to choose the regularizer R that complies with the desired quantization requirements, i.e. satisfying requirement (2). In this section, we present a principled view of *Wasserstein regularization* and show that it is able to derive regularizers for binary, ternary, and multi-bit quantization. Apart from satisfying requirement (2), Wasserstein regularizers often have a simple closed-form prox operator, which makes it suitable for prox-gradient training.

Recall that the p-Wasserstein distance ($p \in [1, \infty]$) between any two distributions P, Q on $\mathbb{R}$ is defined as

$$W_p(P, Q) = \left(\inf_{\pi\in\Pi(P,Q)} \mathbb{E}_{(X,Y)\sim\pi}[|X - Y|^p] \right)^{1/p}, \tag{6}$$

where $\Pi(P, Q)$ is the set of all possible couplings of P and Q, i.e. distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are P and Q.

3.1.1 Wasserstein regularizers for binary, ternary, and multi-bit quantization

Quantization can be viewed as a constraint on the empirical distribution of θ: we would like the empirical distribution of the entries of θ (denoted as $\mathcal{L}(\theta)$) be in a desired set $\mathcal{P}$. This constraint can be encouraged by taking the regularizer as the minimal distance of the empirical distribution of the entries of θ to $\mathcal{P}$ in the Wasserstein metric:

$$R(\theta) = \inf_{P\in\mathcal{P}} W(\mathcal{L}(\theta), P), \tag{7}$$

where W is a Wasserstein distance or a power of Wasserstein distance. We will focus on $W \in \{W_1, W_2^2\}$. As we will see, choosing $W = W_1$ leads to a non-smooth regularizer that induces

exact quantizedness in the same way that L_1 regularization induces sparsity [19], whereas choosing $W = W_2^2$ has an effect similar to L_2 regularization.

We now derive Wasserstein regularizers and the corresponding prox operators for binary, ternary, and multi-bit quantization.

Binary neural nets In a binary neural net, the entries of θ are in $\{\pm 1\}$. Hence the empirical distribution of the weights is a Bernoulli distribution on $\{\pm 1\}$ with some probability p, and so with the choice $W = W_1$ we have

$$R_{\text{bin}}(\theta) = \inf_{p \in [0,1]} W_1(\mathcal{L}(\theta), \text{Ber}(p)).$$

This term has a simpler expression. Indeed, by definition (6), we have

$$R_{\text{bin}}(\theta) = \inf_{p \in [0,1]} \inf_{\pi \in \Pi(\mathcal{L}(\theta), \text{Ber}(p))} \mathbb{E}_{(X,Y)\sim\pi}[|X - Y|].$$

Swapping the two infs, we are allowed to choose any coupling of $\mathcal{L}(\theta)$ and $\{\pm 1\}$, and the optimal coupling has to be mapping positives to one and negatives to minus one. So we have

$$R_{\text{bin}}(\theta) = \frac{1}{d} \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\}. \tag{8}$$

Modulo scaling, this is exactly the binary regularizer (3), and we now see that it originates nicely from a Wasserstein regularization perspective. Figure 2 plots the W-shaped one-dimensional component of R_{bin} from which we see its analog to the L_1 regularization for inducing exact sparsity.

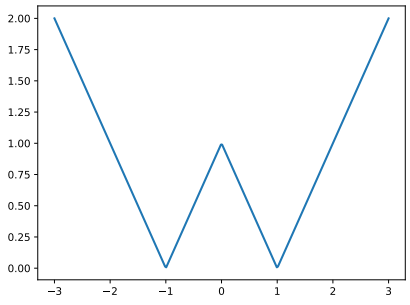


Figure 2: Non-smooth W_1 regularizer for binary quantization (in one dimension): $R(x) = \min\{|x - 1|, |x + 1|\}$.

The prox operator with respect to R_{bin} (ignoring the $1/d$ factor) has a simple form:

$$\begin{aligned} \text{prox}_{\lambda R_{\text{bin}}}(\theta) &= \mathsf{SoftThreshold}(\theta, \text{sign}(\theta), \lambda) \\ &= \text{sign}(\theta) + \text{sign}(\theta - \text{sign}(\theta)) \odot [|\theta - \text{sign}(\theta)| - \lambda]_+. \end{aligned} \tag{9}$$

We note that the choice of the $W = W_1$ is not special: the squared 2-Wasserstein distance W_2^2 works as well, whose prox operator is given by $(\theta + \lambda \text{sign}(\theta))/(1 + \lambda)$. See Appendix A.2 for the derivation of these prox operators and Appendix A.3 for an extension into one-bit quantization with scale.

Multi-bit quantization Multi-bit quantization is needed when binary models do not have the enough expressive power. Following [22], we consider k-bit quantized parameters $\theta \in \mathcal{Q}$, where $\mathcal{Q}$ has the form

$$\mathcal{Q} = \left\{ \sum_{i=1}^{k} \alpha_i b_i : \{\alpha_1, \ldots, \alpha_k\} \subset \mathbb{R},\ b_i \in \{\pm 1\}^d \right\}. \tag{10}$$

In this case, we take the Wasserstein regularizer as

$$R(\theta) = \inf_{\theta_0 \in \mathcal{Q}} W(\mathcal{L}(\theta), \mathcal{L}(\theta_0)). \tag{11}$$

As it is difficult to obtain the closed-form solution for θ_0, we adopt an approximation inspired by (9): find a quantized $\widehat{\theta} \in \mathcal{Q}$ that is close to θ, and take an approximate prox operator as (for $W = W_1$ and $W = W_2^2$ respectively)

$$\mathrm{prox}_{\lambda R}(\theta) = \begin{cases} \mathsf{SoftThreshold}(\theta, \widehat{\theta}, \lambda) & \text{for } W = W_1; \\ \dfrac{\theta + \lambda\widehat{\theta}}{1+\lambda} & \text{for } W = W_2^2. \end{cases} \tag{12}$$

In the case of binary quantization and $\mathsf{q}(\theta) = \mathrm{sign}(\theta)$, this approximation is exactly (9) (and the following W_2^2 version), and in the multi-bit case is a strategy to avoid solving (11) exactly. Further, one has the degree of freedom to choose the implementation of $\widehat{\theta} = \mathsf{q}(\theta)$, but a good choice would be to balance the trade-off between computational efficiency and quantization accuracy.

Ternary quantization Ternary quantization is a variant of 2-bit quantization, in which weights are constrained to be in $\{-\alpha, 0, \beta\}$ for real values $\alpha, \beta > 0$. For ternary quantizaiton, we use the same approximate prox operator (12), but with a ternary quantizer $\widehat{\theta} = \mathsf{q}(\theta)$ defined as

$$\mathsf{q}(\theta) = \theta^+ \mathbf{1}\{\theta \geq \Delta\} + \theta^- \mathbf{1}\{\theta \leq -\Delta\}, \quad \Delta = \frac{0.7}{d}\|\theta\|_1, \quad \theta^+ = \overline{\theta|_{i:\theta_i \geq \Delta}}, \quad \theta^- = \overline{\theta|_{i:\theta_i \leq -\Delta}}. \tag{13}$$

This is a straightforward extension of the TWN quantizer [14] that allows different levels for positives and negatives.

For both ternary and multi-bit quantization, we use the W_2^2 version of the approximate prox operator (i.e. averaging).

3.2 Homotopy method for regularization strength

We now discuss the choice of λ_t, the regularization strength. Recall that the larger λ_t is, the more aggressive θ_{t+1} will move towards the quantized set. An ideal choice would be to (1) force the net to be exactly quantized upon convergence, and (2) not be too aggressive such that the quantized net at convergence is sub-optimal. This suggests that the regularization strength should increase over time.

We let λ_t to be a linearly increasing sequence, i.e. $\lambda_t := \lambda \cdot t$ for some hyper-parameter $\lambda > 0$ which we term as the *regularization rate*. With this choice, the stochastic gradient steps will start off close to full-precision training and gradually move towards exact quantizedness, hence the name "homotopy method". The parameter λ can be tuned by minimizing the validation loss, and controls the aggressiveness of falling onto the quantization constraint. We note that nothing special about the linear increasing scheme, but it is simple enough and works well as we shall see in the experiments.

4 Experiments

We evaluate the performance of ProxQuant on two tasks: image classification with ResNets, and language modeling with LSTMs. On both tasks, we show that the default straight-through gradient method is not the go-to choice, and our ProxQuant can achieve the same and often better results.

4.1 Image classification on CIFAR-10

Problem setup We perform image classification on the CIFAR-10 dataset, which contains 50000 training images and 10000 test images of size 32x32. We apply a commonly used data augmentation strategy (pad by 4 pixels on each side, randomly crop to 32x32, do a horizontal flip with probability 0.5, and normalize). Our models are ResNets [10] of depth 20, 32, and 44 with the weights in all layers quantized to binary or ternary.

Method We use ProxQuant with regularizer (3) in the binary case and (12)+(13) in the ternary case, which we respectively denote as PQ-B and PQ-T. The training is initialized at pre-trained full-precision nets (warm-start). For the regularization strength we use the homotopy method $\lambda_t = \lambda \cdot t$ with $\lambda = 10^{-4}$. We initialize at pre-trained full-precision networks and use the Adam optimizer with constant learning rate 0.01. As the prox operator does not strictly enforce quantizedness, to accelerate training in the final stage, we do a hard quantization $\theta \mapsto \mathsf{q}(\theta)$ at epoch 400 and keeps training till the 600-th epoch to stabilize the BatchNorm layers.

We compare with BinaryConnect (BC) for binary nets and Trained Ternary Quantization (TTQ) [24] for ternary nets. For BinaryConnect, we haven't found reported results with ResNets on CIFAR-10, and we train with the recommended Adam optimizer with learning rate decay [5] (initial learning rate 0.01, multiply by 0.1 at epoch 81 and 122, hard-quantize at epoch 400), which we find leads to the best result for BinaryConnect.

Result The top-1 classification errors are reported in Table 1. For binary nets, our ProxQuant-Binary consistenly yields better results than BinaryConnect. For ternary nets, our results are on par with the reported results of Trained Ternary Quantization.[1]

Table 1: Top-1 classification error of quantized ResNets on CIFAR-10. Performance is reported in mean(std) over 4 runs, where for PQ-T we report in addition the best of 4.

Model (Bits)	Full-Precision (32)	BC (1)	PQ-B (ours) (1)	TTQ (2)	PQ-T (ours) (2)	PQ-T (best of 4) (2)
ResNet-20	8.06	9.49 (0.22)	**9.15** (0.21)	8.87	**8.37** (0.05)	8.32
ResNet-32	7.25	8.66 (0.36)	**8.40** (0.23)	7.63	7.63 (0.11)	7.55
ResNet-44	6.96	8.26 (0.24)	**7.79** (0.06)	7.02	7.11 (0.13)	7.00

4.2 Language modeling with LSTMs

Problem setup We perform language modeling tasks with LSTMs [11] on the Penn Treebank (PTB) dataset [16], which contains 929K training tokens, 73K validation tokens, and 82K test tokens. Our model is a standard one-hidden-layer LSTM with embedding dimension 300 and hidden dimension 300. We train quantized LSTMs with the encoder, transition matrix, and the decoder quantized to k-bits for $k \in \{1, 2, 3\}$. The quantization is performed in a row-wise fashion, so that each row of the matrix has its own codebook $\{\alpha_1, \ldots, \alpha_k\}$. We use the alternating multi-bit (ALT) quantizer [22], a powerful multi-bit quantization algorithm giving quantization of the form (10).

[1] We note that our ProxQuant-Ternary and TTQ are not in a strict sense comparable: we have the advantage of using better initializations; TTQ has the advantage of a stronger quantizer: they train the quantization levels (θ^+, θ^-) whereas our quantizer (13) pre-computes them from the current full-precision parameter.

Method Using the ALT quantizer, we compare two training dynamics: the straight-through gradient method of [22] and our ProxQuant. Training is initialized at a pre-trained full-precision LSTM. We use the SGD optimizer with initial learning rate 20.0 and decay by a factor of 1.2 when the validation error does not improve over an epoch. We train for 80 epochs with batch size 20, BPTT 30, dropout with probability 0.5, and clip the gradient norms to 0.25. The regularization rate λ is tuned by finding the best performance on the validation set. In addition to multi-bit quantization we also report the results for binary LSTMs (weights in $\{\pm 1\}$), comparing BinaryConnect and our ProxQuant-Binary.

Result We report the perplexity-per-word (lower is better) in Table 2. The performance of ProxQuant is comparable with the Straight-through gradient method and in case of Binary LSTMs beats BinaryConnect by a large margin, demonstrating that ProxQuant is effective for training recurrent networks.

Table 2: PPW of quantized LSTM on Penn Treebank.

Method / Number of Bits	1	2	3	FP (32)
BinaryConnect	419.1	-	-	88.5
ProxQuant-Binary (ours)	321.8	-	-	
Alt + Straight-through[2]	104.7	90.2	86.1	
Alt + ProxQuant (ours)	106.7	90.4	87.8	

5 Stability analysis through sign change

We further compare the training dynamics of ProxQuant-Binary and BinaryConnect through the *sign change* metric. The sign change metric between any θ_1 and θ_2 is the proportion of their different signs, i.e. the (rescaled) Hamming distance:

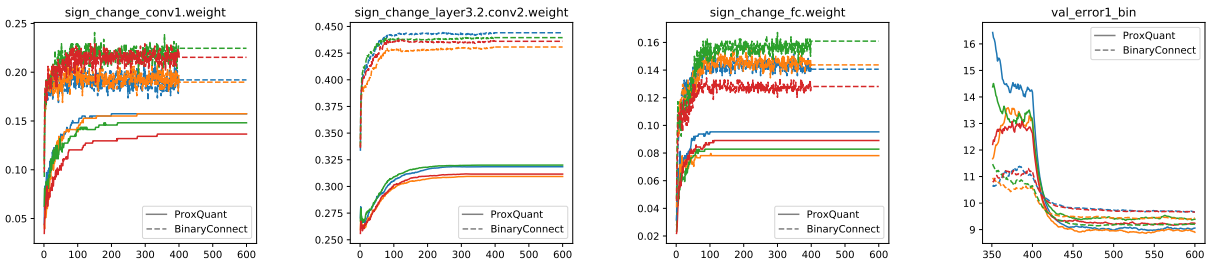
$$\mathsf{SignChange}(\theta_1, \theta_2) = \frac{\|\mathrm{sign}(\theta_1) - \mathrm{sign}(\theta_2)\|_1}{2d} \in [0, 1].$$

In $\mathbb{R}^d$, the space of all full-precision parameters, the sign change is a natural distance metric that represents the closeness of the binarization of two parameters.

Recall in our CIFAR-10 experiments (Section 4.1), for both BinaryConnect and ProxQuant, we initialize at a good full-precision net θ_0 and stop at a converged binary network $\widehat{\theta} \in \{\pm 1\}^d$ (ignoring BatchNorm layers and biases that we don't quantize). We are interested in $\mathsf{SignChange}(\theta_0, \theta_t)$ along the training path, as well as $\mathsf{SignChange}(\theta_0, \widehat{\theta})$, i.e. the distance of the final output model to the initialization.

As ProxQuant converges to higher-performance solutions than BinaryConnect, we expect that if we run both methods from a same warm start, the sign change of ProxQuant should be higher than that of BinaryConnect, as in general one needs to travel farther to find a better net.

[2]We thanks Xu et al. [22] for sharing the implementation of this method through a personal communication. There is a very clever trick not mentioned in their paper: after computing the alternating quantization $\mathsf{q}(\theta)$, they multiply by a constant 0.3 before taking the gradient; in other words, their quantizer is a rescaled ALT: $\theta \mapsto 0.3\mathsf{q}(\theta)$. This scaling trick gives a significant gain in performance – without scaling the PPW is $\{116.7, 94.3, 87.3\}$ for $\{1, 2, 3\}$ bits. In contrast, our ProxQuant uses the unscaled ALT quantizer and achieves slightly better PPW (for 1 and 2 bit) than straight-through with the same unscaled ALT.



(a) (b) (c) (d)

Figure 3: $\mathsf{SignChange}(\theta_0, \theta_t)$ against t (epoch) for BinaryConnect and ProxQuant, over 4 runs starting from the same full-precision ResNet-20. ProxQuant has significantly lower sign changes than BinaryConnect while converging to better models. (a) The first convolutional layer of size $16 \times 3 \times 3 \times 3$; (b) The last convolutional layer of size $64 \times 64 \times 3 \times 3$; (c) The fully connected layer of size 64×10; (d) The validation top-1 error of the binarized nets (with moving average smoothing).

However, we find that this is not the case: ProxQuant produces binary nets with both *lower* sign changes and *higher* performances, compared with BinaryConnect. This finding is consistent in all layers, across different warm starts, and across differnent runs from each same warm start (see Figure 3 and Table 3). This shows that for every warm start position, there is a good binary net nearby which can be found by ProxQuant but not BinaryConnect, suggesting that BinaryConnect, and in general the straight-through gradient method, suffers from higher optimization instability compared with ProxQuant. We argue that this instability is inherent: BinaryConnect has a very stringent convergence condition (Theorem C.1), which is also reflected in the experiment: the signs in BinaryConnect never stop changing until we manually freeze the signs at epoch 400.

6 Conclusion

In this paper, we propose and experiment with the ProxQuant method for training quantized networks. Our results demonstrate that ProxQuant offers a powerful alternative to the straight-through gradient method and suffers from less optimization instability. For future work, it would be of interest to propose alternative regularizers for multi-bit quantization and experiment with our method on larger tasks and networks.

References

[1] A. G. Anderson and C. P. Berg. The high-dimensional geometry of binary neural networks. *arXiv preprint arXiv:1705.07199*, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] M. A. Carreira-Perpinán. Model compression as constrained optimization, with application to neural nets. part i: General framework. *arXiv preprint arXiv:1707.01209*, 2017.

[4] M. A. Carreira-Perpinán and Y. Idelbayev. Model compression as constrained optimization, with application to neural nets. part ii: Quantization. *arXiv preprint arXiv:1707.04319*, 2017.

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

[6] Y. Ding, J. Liu, and Y. Shi. On the universal approximability of quantized relu neural networks. *arXiv preprint arXiv:1802.03646*, 2018.

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

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

[9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. EIE: Efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.

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

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187–1, 2017.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[15] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pages 5811–5821, 2017.

[16] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[18] J. Sun and X. Sun. Adversarial probabilistic regularization. Unpublished draft, 2018.

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[20] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[21] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

[22] C. Xu, J. Yao, Z. Lin, W. Ou, Y. Cao, Z. Wang, and H. Zha. Alternating multi-bit quantization for recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S19dR9x0b`.

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

[24] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

A Additional results on Wasserstein regularizers

A.1 Recap on Wasserstein distance

Given two distributions P, Q in a metric space with distance $d(\cdot, \cdot)$, the p-Wasserstein distance ($p \in [1, \infty]$) is defined as

$$W_p^p(P, Q) = \inf_{\pi \in \Pi(P,Q)} \mathbb{E}_{(X,Y)\sim\pi}[d(X, Y)^p], \tag{14}$$

where Π is the set of all possible couplings of (P, Q). The Kantorovich duality states that the following duality holds for W_1 [20]:

$$W_1(P, Q) = \sup_{\|f\|_{\text{Lip}} \le 1} \mathbb{E}_P[f(X)] - \mathbb{E}_Q[f(X)], \tag{15}$$

where $\|f\|_{\text{Lip}} = \sup_{x \neq y} \frac{f(x)-f(y)}{d(x,y)}$ is the Lipschitz constant of f (with respect to d).

A.2 Prox operators for binary nets

Here we derive the prox operators for the binary regularizer (8) and its W_2^2 variant. Recall that (ignoring the $1/d$ scaling)

$$R_{\text{bin}}(\theta) = \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\}.$$

By definition of the prox operator, we have for any $\theta^0 \in \mathbb{R}^d$ that

$$\begin{aligned}\text{prox}_{\lambda R_{\text{bin}}}(\theta^0) &= \arg\min_{\theta \in \mathbb{R}^d} \left\{ \frac{1}{2} \left\|\theta - \theta^0\right\|_2^2 + \lambda \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\} \right\} \\ &= \arg\min_{\theta \in \mathbb{R}^d} \left\{ \sum_{j=1}^{d} \frac{1}{2}(\theta_j - \theta_j^0)^2 + \lambda \min\{|\theta_j - 1|, |\theta_j + 1|\} \right\}.\end{aligned}$$

This minimization problem is coordinate-wise separable. For each θ_j, the penalty term remains the same upon flipping the sign, but the quadratic term is smaller when $\text{sign}(\theta_j) = \text{sign}(\theta_j^0)$. Hence, the solution $\theta^\star$ satisfies that $\text{sign}(\theta_j^\star) = \text{sign}(\theta_j^0)$, and the absolute value satisfies

$$|\theta_j^\star| = \arg\min_{t \ge 0} \left\{ \frac{1}{2}(t - |\theta_j^0|)^2 + \lambda |t - 1| \right\} = \mathsf{SoftThreshold}(|\theta_j^0|, 1, \lambda) = 1 + \text{sign}(|\theta_j^0| - 1)[||\theta_j^0| - 1| - \lambda]_+.$$

Multiplying by $\mathrm{sign}(\theta_j^\star) = \mathrm{sign}(\theta_j^0)$, we have

$$\theta_j^\star = \mathsf{SoftThreshold}(\theta_j^0, \mathrm{sign}(\theta_j^0), \lambda),$$

which gives (9).

For $W = W_2^2$, the optimal coupling to Bernoulli distributions is identical to the W_1 case: map each entry to its sign. The corresponding Wasserstein regularizer is

$$R_{\mathrm{bin}}(\theta) = d \cdot \inf_{p\in[0,1]} W_2^2(\mathcal{L}(\theta), \mathrm{Ber}(p)) = \sum_{j=1}^d \min\left\{(\theta_j - 1)^2, (\theta_j + 1)^2\right\}.$$

For this regularizer we have

$$\mathrm{prox}_{\lambda R_{\mathrm{bin}}}(\theta^0) = \arg\min_{\theta\in\mathbb{R}^d} \left\{ \sum_{j=1}^d \frac{1}{2}(\theta_j - \theta_j^0)^2 + \lambda \min\left\{(\theta_j - 1)^2, (\theta_j + 1)^2\right\} \right\}.$$

Using the same argument as in the W_1 case, the solution $\theta^\star$ satisfies $\mathrm{sign}(\theta_j^\star) = \mathrm{sign}(\theta_j^0)$, and

$$|\theta_j^\star| = \arg\min_{t\geq 0} \left\{ \frac{1}{2}(t - |\theta_j^0|)^2 + \lambda(t-1)^2 \right\} = \frac{|\theta_j^0| + \lambda}{1+\lambda}.$$

Multiplying by $\mathrm{sign}(\theta_j^\star) = \mathrm{sign}(\theta_j^0)$ gives

$$\theta_j^\star = \frac{\theta_j^0 + \lambda \mathrm{sign}(\theta_j^0)}{1+\lambda},$$

or, in vector form, $\theta^\star = (\theta + \lambda \mathrm{sign}(\theta))/(1+\lambda)$.

A.3 One-bit quantization with scale

A straightforward extension for binary nets is to allow for a (learnable) scale, that is, a quantized parameter is $\theta = \alpha b$, where $\alpha \geq 0$ and $b \in \{\pm 1\}^d$. Here the set of quantized empirical distributions is $\mathcal{P} = \{\alpha \cdot \mathrm{Ber}(p) : \alpha \geq 0, p \in [0,1]\}$, and the Wasserstein regularizer has the form

$$R(\theta) = \inf_{\alpha\geq 0, p\in[0,1]} W(\mathcal{L}(\theta), \alpha \cdot \mathrm{Ber}(p)).$$

It turns out that this regularizer has a closed-form expression:

Lemma A.1. *Let $\alpha_\star = \alpha_\star(\theta)$ be the α that achieves the infimum above. We have*

$$\begin{aligned} &\alpha_\star = \mathsf{med}(|\theta|), \quad R(\theta) = \frac{1}{d}\sum_{j=1}^d \min\left\{|\theta_j - \alpha_\star|, |\theta_j + \alpha_\star|\right\} \quad \text{for} \quad W = W_1; \\ &\alpha_\star = \overline{|\theta|} = \frac{1}{d}\left\|\theta\right\|_1, \quad R(\theta) = \frac{1}{d}\sum_{j=1}^d \min\left\{(\theta_j - \alpha_\star)^2, (\theta_j + \alpha_\star)^2\right\} \quad \text{for} \quad W = W_2^2. \end{aligned} \tag{16}$$

To compute its prox requires differentiating $\alpha_\star$ w.r.t. θ. We here take an approximation that ignores this differentiation and treats $\alpha_\star$ as fixed: this gives the approximation

$$\mathrm{prox}_{\lambda R_{\mathrm{bin}}}(\theta) = \begin{cases} \mathsf{SoftThreshold}(\theta, \alpha_\star \mathrm{sign}(\theta), \lambda) & \text{for } W = W_1 \\ \dfrac{\theta + \lambda\alpha_\star \mathrm{sign}(\theta)}{1+\lambda} & \text{for } W = W_2^2. \end{cases}$$

Proof of Lemma A.1 We first consider the $W = W_1$ case. For each $\alpha \geq 0$, we have that

$$\inf_{p\in[0,1]} W(\mathcal{L}(\theta), \alpha \cdot \mathrm{Ber}(p)) = \frac{1}{d}\sum_{j=1}^{d} \min\left\{|\theta_j - \alpha|, |\theta_j + \alpha|\right\} = \frac{1}{d}\sum_{j=1}^{d} \left||\theta_j| - \alpha\right|.$$

As α varies, the minimizing α of the above quantity is $\alpha^\star = \mathsf{med}\{|\theta_1|, \ldots, |\theta_d|\}$, which is the desired result. For $W = W_2^2$, the corresponding quantity is $\frac{1}{d}\sum_{j=1}^{d}(|\theta_j| - \alpha)^2$, whose minimizer is $\alpha_\star = \frac{1}{d}\sum_{j=1}^{d}|\theta_j| = \frac{1}{d}\|\theta\|_1$. □

B Detailed sign change results on ResNet-20

Table 3: Performances and sign changes on ResNet-20 in mean(std) over 3 full-precision initializations and 4 runs per (initialization x method). Sign changes are computed over all quantized parameters in the net.

Initialization	Method	Top-1 Error(%)	Sign change
FP-Net 1 (8.06)	BC	9.489 (0.223)	0.383 (0.006)
	PQ-B	**9.146** (0.212)	**0.276** (0.020)
FP-Net 2 (8.31)	BC	9.745 (0.422)	0.381 (0.004)
	PQ-B	**9.444** (0.067)	**0.288** (0.002)
FP-Net 3 (7.73)	BC	9.383 (0.211)	0.359 (0.001)
	PQ-B	**9.084** (0.241)	**0.275** (0.001)

Table 4: Performances and sign changes on ResNet-20 in raw data over 3 full-precision initializations and 4 runs per (initialization x method). Sign changes are computed over all quantized parameters in the net.

Initialization	Method	Top-1 Error(%)	Sign change
FP-Net 1 (8.06)	BC	9.664, 9.430, 9.198, 9.663	0.386, 0.377, 0.390, 0.381
	PQ-B	9.058, 8.901, 9.388, 9.237	0.288, 0.247, 0.284, 0.285
FP-Net 2 (8.31)	BC	9.456, 9.530, 9.623, 10.370	0.376, 0.379, 0.382, 0.386
	PQ-B	9.522, 9.474, 9.410, 9.370	0.291, 0.287, 0.289, 0.287
FP-Net 3 (7.73)	BC	9.107, 9.558, 9.538, 9.328	0.360, 0.357, 0.359, 0.360
	PQ-B	9.284, 8.866, 9.301, 8.884	0.275, 0.276, 0.276, 0.275

C Characterization of fixed points of BinaryConnect

Consider the BinaryConnect method with batch gradient descent:

$$\begin{aligned} \theta_t &= \mathrm{sign}(h_t) \\ h_{t+1} &= h_t - \eta_t \nabla L(\theta_t). \end{aligned} \tag{17}$$

Definition C.1 (Fixed points and the convergence)**.** *We say that $\theta \in \{-1,1\}^d$ is a **fixed point** of the BinaryConnect algorithm, if there exists $h \in \mathbb{R}^d$ such that when we take $h_0 = h$ in the iterates* (17) *$\theta_t = \theta$ for all $t = 0, 1, 2, ...$ We say that the BinaryConnect algorithm **converges** with an initialization $h_0 \in \mathbb{R}^d$ if there exists $t \in \infty$ such that θ_t is a fixed point.*

Theorem C.1. *Assume that the sequence of nonnegative learning rates obeys that $\lim_{T\to\infty} \sum_{t=0}^{T} \eta_t = \infty$, then $\theta \in \{\pm 1\}^d$ is a fixed point for BinaryConnect* (17) if and only if *$sign(\nabla L(\theta)[i]) = -\theta[i]$ for all $i \in [d]$ such that $\nabla L(\theta)[i] \neq 0$. Such a point may not exist, in which case BinaryConnect does not converge with any initialization $h_0 \in \mathbb{R}^d$.*

Proof. We start with the "$\Rightarrow$" direction. If θ is a fixed point, then by definition there exists $h_0 \in \mathbb{R}^d$ such that $\theta_t = \theta$ for all $t = 0, 1, 2, ...$ By the iterates (17)

$$h_T = h_0 - \sum_{t=0}^{T} \eta_t \nabla L(\theta_t).$$

Take signs on both sides and apply $\theta_t = \theta$ for all t on both sides, we get that

$$\theta = \theta_T = \text{sign}(h_T) = \text{sign}\left(h_0 - \nabla L(\theta) \sum_{t=0}^{T} \eta_t \right)$$

Take the limit $T \to \infty$ and apply the assumption that $\sum_t \eta_t = \infty$, we get that for all $i \in [d]$ such that $[\nabla L(\theta)]_i \neq 0$,

$$\theta[i] = \lim_{T\to\infty} \text{sign}\left(h_0 - \nabla L(\theta) \sum_{t=0}^{T} \eta_t \right)[i] = -\text{sign}(\nabla L(\theta))[i].$$

Now we prove the "$\Leftarrow$" direction. If θ obeys that $\text{sign}(\nabla L(\theta)[i]) = -\theta[i]$ for all $i \in [d]$ such that $\nabla L(\theta)[i] \neq 0$, then if we take any h_0 such that $\text{sign}(h_0) = \theta$, h_t will move in a straight line towards the direction of $-\nabla L(\theta)$, which does not change the sign of h_0. In other word, $\theta_t = \text{sign}(h_t) = \text{sign}(h_0) = \theta$ for all $t = 0, 1, 2, ...$ Therefore, by definition, θ is a fixed point.

Lastly, to see that such a fixed point θ might not exists in general, see the example in Figure 1b. □