ProxQuant: Quantized Neural Networks via Proximal Operators

Anonymous Author(s)
Affiliation
Address
email

Abstract

To make deep neural networks feasible in resource-constrained environments (such as mobile devices), it is beneficial to quantize models by using low-precision weights. One common technique for quantizing neural networks is the straight-through gradient method, which enables back-propagation through the quantization mapping. Despite its empirical success, little is understood about why the straight-through gradient method works.

Building upon a novel observation that the straight-through gradient method is in fact *identical* to the well-known Nesterov's dual-averaging algorithm on a quantization constrained optimization problem, we propose a more principled alternative approach, called ProxQuant, that formulates quantized network training as a regularized learning problem instead and optimizes it 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. For binary quantization, our analysis shows both theoretically and experimentally that ProxQuant is more stable than the straight-through gradient method (i.e. BinaryConnect), challenging the indispensability of the straight-through gradient method and providing a powerful alternative.

1 Introduction

In this paper, we formulate the problem of model quantization as a regularized learning problem and propose to solve it with a proximal gradient method. Our contributions are summarized as follows.

- We present a unified framework for defining regularization functionals that encourage binary, ternary, and multi-bit quantized parameters, through penalizing the distance to quantized sets. For binary quantization, the resulting regularizer is a W-shaped non-smooth regularizer, which shrinks parameters towards either -1 or 1 in the same way that the L_1 norm regularization shrinks parameters towards 0. We demonstrate that the prox-operators for regularizers that come out of our framework often admit linear-time solutions (or linear time approximation heuristics) which result in numerically *exact* quantized parameters.
- We propose training quantized networks using ProxQuant (Algorithm 1) — a stochastic proximal gradient method with a homotopy scheme. Compared with the straight-through gradient method, ProxQuant has access to additional gradient information at non-quantized points, and its homotopy scheme prevents potential overshoot early in the training. Algorithmically, ProxQuant involves just adding a simple proximal step with respect to a quantization-inducing regularizer after each stochastic gradient step, thus can be efficiently

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

implemented under any major deep learning frameworks without incurring significant system overhead and be used as a modular component to add to the training pipeline of any deep networks to result in a quantized network.

- We demonstrate the effectiveness and flexibility of ProxQuant through systematic experiments on (1) image classification with ResNets (Section 3.1); (2) language modeling with LSTMs (Section 3.2). The ProxQuant method outperforms the state-of-the-art results on binary quantization and is comparable with the state-of-the-art on ternary and multi-bit quantization.
- For binary nets, we show that BinaryConnect suffers from more optimization instability than ProxQuant through (1) a theoretical characterization of convergence for BinaryConnect (Section 4.1) and (2) a sign change experiment on CIFAR-10 (Section G). Experimentally, ProxQuant finds better binary nets that is also closer to the initialization in the sign change metric.

We present the main ingredients of our contribution in this extended abstract. See the Appendices B for the prior work, C for the notation, A and D for the motivation and preliminary discussions about the straight-through gradient method and prox operators.

2 Quantized net training via regularized learning

We propose the ProxQuant algorithm, which adds a quantization-inducing regularizer onto the loss and optimizes via the (non-lazy) prox-gradient method with 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 induces 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

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

$$= \mathrm{prox}_{\eta_t \lambda_t R} \left(\theta_t - \eta_t \widetilde{\nabla} L(\theta_t) \right). \tag{2}$$

The inner SGD step in eq. (2) can be replaced by any preferred stochastic optimization method such as Momentum SGD or Adam [Kingma and Ba, 2014].

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 learning rates 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.

Details of choosing the regularizer R, deriving the prox-operator $\mathrm{prox}_{\lambda R}$, and choosing the regularization strength are deferred to Appendix E.

3 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 only choice, and our ProxQuant can achieve the same and often better results.

3.1 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 [He et al., 2016] of depth 20, 32, and 44 with weights quantized to binary or ternary.

Method We use ProxQuant with suitable regularizers in the binary case and 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. 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) [Zhu et al., 2016] 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 [Courbariaux et al., 2015] (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 consistently yields better results than BinaryConnect. For ternary nets, our results are comparable with the reported results of TTQ, and the best performance of our method over 4 runs (from the same initialization) is slightly better than TTQ.

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 (Bo4).

Model (Bits)	Full-Precision (32)	BC (1)	PQ-B (ours) (1)	TTQ (2)	PQ-T (ours) (2)	PQ-T (Bo4) (2)
ResNet-20	8.06	9.49 (0.22)	**9.15** (0.21)	8.87	**8.40** (0.13)	8.22
ResNet-32	7.25	8.66 (0.36)	**8.40** (0.23)	7.63	7.65 (0.15)	7.53
ResNet-44	6.96	8.26 (0.24)	**7.79** (0.06)	7.02	7.05 (0.08)	6.98

3.2 Language modeling with LSTMs

See Appendix F for details.

4 Stability analysis of binary quantization

4.1 Convergence characterization for BinaryConnect

We now show that BinaryConnect has a very stringent convergence condition. Consider the BinaryConnect method with batch gradients:

$$s_t = \text{sign}(\theta_t), \quad \theta_{t+1} = \theta_t - \eta_t \nabla L(s_t). \tag{3}$$

Definition 4.1 (Fixed point and convergence)**.** *We say that $s \in \{\pm 1\}^d$ is a **fixed point** of the BinaryConnect algorithm, if $s_0 = s$ in eq.* (3) *implies that $s_t = s$ for all $t = 1, 2, ...$. We say that the BinaryConnect algorithm **converges** if there exists $t < \infty$ such that s_t is a fixed point.*

Theorem 4.1. *Assume that the learning rates satisfy $\sum_{t=0}^{\infty} \eta_t = \infty$, then $s \in \{\pm 1\}^d$ is a fixed point for BinaryConnect eq.* (3) if and only if *$\text{sign}(\nabla L(s)[i]) = -s[i]$ 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 for any initialization $\theta_0 \in \mathbb{R}^d$.*

In a sign change experiment on CIFAR-10 (see Appendix G), we are going to see that BinaryConnect indeed fails to converge to a fixed sign pattern, corroborating Theorem 4.1.

References

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

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

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

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.

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.

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

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.

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.

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.

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

L. Hou and J. T. Kwok. Loss-aware weight quantization of deep networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkrSv0lA-`.

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.

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

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

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.

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.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3): 127–239, 2014.

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.

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

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

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

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`.

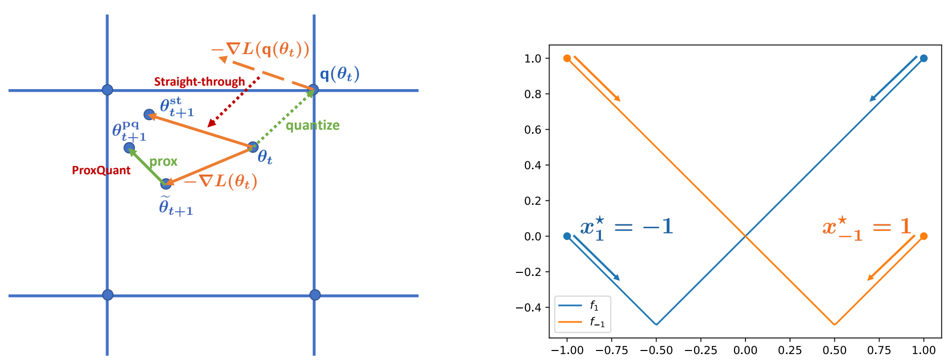
S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

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

A Drawback of the straight-through gradient method

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* [Courbariaux et al., 2015] 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 $\theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\theta)|_{\theta=\mathsf{q}(\theta_t)}$, 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 [Courbariaux et al., 2015].

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 necessarily contain any 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.

B Prior work

Methodologies Han et al. [2015] 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 [Han et al., 2016]. In a parallel line of work, Courbariaux et al. [2015] propose BinaryConnect that enables the training of binary neural networks, and Li and Liu [2016], Zhu et al. [2016] extend this method into ternary quantization. Training and inference on quantized nets can be made more efficient by also quantizing the activation [Hubara et al., 2017, Rastegari et al., 2016, Zhou et al., 2016], and such networks have achieved impressive performance on large-scale tasks such as ImageNet classification [Rastegari et al., 2016, Zhu et al., 2016]. In the NLP land, quantized language models have been successfully trained using alternating multi-bit quantization [Xu et al., 2018].

Theories Li et al. [2017] 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 [2017] 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. [2018] show a universal approximation theorem for quantized ReLU networks.

Principled methods Hou and Kwok [2018] propose a proximal Newton algorithm for model quantization, which makes use of the additional Hessian information but is hence slightly more expensive in each iteration. Sun and Sun [2018] perform model quantization through a Wasserstein regularization term and minimize via the adversarial representation, similar as in Wasserstein GANs [Arjovsky et al., 2017]. Their method has the potential of generalizing to other generic requirements on the parameter, but might be hard to tune due to the instability of the inner maximization problem.

While preparing this manuscript, we discovered the independent work of Carreira-Perpinán [2017], Carreira-Perpinán and Idelbayev [2017]. They formulate quantized network training as a constrained optimization problem and propose to solve them via augmented Lagrangian methods. 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.

C 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 Appendix D.3). The p-Wasserstein distance is denoted as W_p. We will restrict attention to Wasserstein distances on $\mathbb{R}$ and $p \in \{1, 2\}$.

D Preliminaries

The optimization difficulty of 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 [Li et al., 2017], 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.

D.1 The straight-through gradient method

The pioneering work of BinaryConnect [Courbariaux et al., 2015] proposes to solve this problem via the *straight-through gradient method*, 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

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

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 has thus become a standard practice [Courbariaux et al., 2015, Zhu et al., 2016, Xu et al., 2018]. However, as we have discussed, it is information theoretically unclear how the straight-through method works, and it does fail on very simple convex Lipschitz functions (Figure 1b).

D.2 Straight-through gradient as lazy projection

Our first observation is that the straight-through gradient method is equivalent to Nesterov's *dual-averaging* method, or a lazy projected SGD [Xiao, 2010]. In the binary case, we wish to minimize $L(\theta)$ over $\mathcal{Q} = \{\pm 1\}^d$, and the lazy projected SGD proceeds as

$$\begin{cases} \widetilde{\theta}_t = \mathrm{Proj}_{\mathcal{Q}}(\theta_t) = \mathrm{sign}(\theta_t) = \mathsf{q}(\theta_t), \\ \theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\widetilde{\theta}_t). \end{cases} \tag{4}$$

Written compactly, this is $\theta_{t+1} = \theta_t - \eta_t \widetilde{\nabla} L(\theta)|_{\theta=\mathsf{q}(\theta_t)}$, which is exactly the straight-through gradient method: take the gradient at the quantized vector and perform the update on the original real vector.

D.3 Projection as a limiting proximal operator

We take a broader point of view that a projection is also a limiting 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 $R : \mathbb{R}^d \to \mathbb{R}_{\ge 0}$ such that the following hold:

$$R(\theta) = 0,\ \ \forall \theta \in \mathcal{Q} \quad \text{and} \quad R(\theta) > 0,\ \ \forall \theta \notin \mathcal{Q}. \tag{5}$$

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

$$R(\theta) = R_{\mathrm{bin}}(\theta) = \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\}. \tag{6}$$

The proximal operator (or prox operator) [Parikh and Boyd, 2014] with respect to R and strength $\lambda > 0$ is

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

In the limiting case $\lambda = \infty$, 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, a finite λ 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.

E Details on the ProxQuant algorithm

E.1 Regularization for model quantization

We define a flexible class of quantization-inducing regularizers through "distance to the quantized set", derive efficient algorithms of their corresponding prox operator, and propose a homotopy method for choosing the regularization strengths. Our regularization perspective subsumes most existing algorithms for model-quantization (e.g.,[Courbariaux et al., 2015, Han et al., 2015, Xu et al., 2018]) as limits of certain regularizers with strength $\lambda \to \infty$. Our proposed method can be viewed as a principled generalization of these methods to $\lambda < \infty$.

Let $\mathcal{Q} \subset \mathbb{R}^d$ be a set of quantized parameter vectors. An ideal regularizer for quantization would be to vanish on $\mathcal{Q}$ and reflect some type of distance to $\mathcal{Q}$ when $\theta \notin \mathcal{Q}$. To achieve this, we propose L_1 and L_2 regularizers of the form

$$R(\theta) = \inf_{\theta_0 \in \mathcal{Q}} \|\theta - \theta_0\|_1 \quad \text{or} \quad R(\theta) = \inf_{\theta_0 \in \mathcal{Q}} \|\theta - \theta_0\|_2^2 . \tag{7}$$

This is a highly flexible framework for designing regularizers, as one could specify any $\mathcal{Q}$ and choose between L_1 and L_2. Specifically, $\mathcal{Q}$ encodes certain desired quantization structure. By appropriately choosing $\mathcal{Q}$, we can specify which part of the parameter vector to quantize[1], the number of bits to quantize to, whether we allow adaptively-chosen quantization levels and so on.

The choice of distance metrics will result in distinct properties in the regularized solutions. For example, choosing the L_1 version leads to non-smooth regularizers that induce exact quantizedness in the same way that L_1 norm regularization induces sparsity [Tibshirani, 1996], whereas choosing the squared L_2 version leads to smooth regularizers that induce quantizedness "softly".

In the following, we present a few examples of regularizers under our framework eq. (7) which induce binary weights, ternary weights and multi-bit quantization. We will also derive efficient algorithms (or approximation heuristics) for solving the prox operators corresponding to these regularizers, which generalize the projection operators used in the straight-through gradient algorithms.

Binary neural nets In a binary neural net, the entries of θ are in $\{\pm 1\}$. A natural choice would be taking $\mathcal{Q} = \{-1, 1\}^d$. The resulting L_1 regularizer is

$$\begin{aligned} R(\theta) &= \inf_{\theta_0 \in \{\pm 1\}^d} \|\theta - \theta_0\|_1 = \sum_{j=1}^{d} \inf_{[\theta_0]_j \in \{\pm 1\}} |\theta_j - [\theta_0]_j| \\ &= \sum_{j=1}^{d} \min\{|\theta_j - 1|, |\theta_j + 1|\} = \|\theta - \mathrm{sign}(\theta)\|_1 . \end{aligned} \tag{8}$$

This is exactly the binary regularizer $R_{\rm bin}$ that we discussed earlier in eq. (6). Figure 2 plots the W-shaped one-dimensional component of $R_{\rm bin}$ from which we see its effect for inducing $\{\pm 1\}$ quantization in analog to L_1 regularization for inducing exact sparsity.

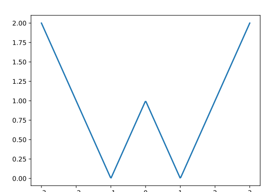


Figure 2: W-shaped regularizer for binary quantization.

The prox operator with respect to $R_{\rm bin}$, despite being a non-convex optimization problem, admits a simple analytical solution:

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

We note that the choice of the L_1 version is not unique: the squared L_2 version works as well, whose prox operator is given by $(\theta + \lambda \mathrm{sign}(\theta))/(1+\lambda)$.

Multi-bit quantization with adaptive levels. Following [Xu et al., 2018], we consider k-bit quantized parameters with a structured adaptively-chosen set of quantization levels, which translates into

$$\mathcal{Q} = \left\{ \sum_{i=1}^{k} \alpha_i b_i : \{\alpha_1, \dots, \alpha_k\} \subset \mathbb{R},\ b_i \in \{\pm 1\}^d \right\} = \left\{ \theta_0 = B\alpha : \alpha \in \mathbb{R}^k,\ B \in \{\pm 1\}^{d \times k} \right\}. \tag{10}$$

[1] Empirically, it is advantageous to keep the biases of each layers and the BatchNorm layers at full-precision, which is often a negligible fraction, say $1/\sqrt{d}$ of the total number of parameters

The squared L_2 regularizer for this structure is

$$R_{k-\mathrm{bit}}(\theta) = \inf_{\alpha\in\mathbb{R}^k, B\in\{\pm1\}^{d\times k}} \left\|\theta - B\alpha\right\|_2^2, \tag{11}$$

which is also the alternating minimization objective in [Xu et al., 2018].

We now derive the prox operator for the regularizer eq. (11). For any θ, we have

$$\begin{aligned} \mathrm{prox}_{\lambda R_{k-\mathrm{bit}}}(\theta) &= \arg\min_{\widetilde{\theta}} \left\{ \frac{1}{2}\left\|\widetilde{\theta}-\theta\right\|_2^2 + \lambda \inf_{\alpha\in\mathbb{R}^k, B\in\{\pm1\}^{d\times k}} \left\|\widetilde{\theta} - B\alpha\right\|_2^2 \right\} \\ &= \arg\min_{\widetilde{\theta}} \inf_{\alpha\in\mathbb{R}^k, B\in\{\pm1\}^{d\times k}} \left\{ \frac{1}{2}\left\|\widetilde{\theta}-\theta\right\|_2^2 + \lambda\left\|\widetilde{\theta} - B\alpha\right\|_2^2 \right\}. \end{aligned} \tag{12}$$

This is a joint minimization problem in $(\widetilde{\theta}, B, \alpha)$, and we adopt an alternating minimization schedule to solve it:

(1) Minimize over $\widetilde{\theta}$ given (B, α), which has a closed-form solution $\widetilde{\theta} = \frac{\theta+2\lambda B\alpha}{1+2\lambda}$.

(2) Minimize over (B, α) given $\widetilde{\theta}$, which does not depend on θ_0, and can be done via calling the alternating quantizer of [Xu et al., 2018]: $B\alpha = \mathsf{q}_{\mathrm{alt}}(\widetilde{\theta})$.

Together, the prox operator generalizes the alternating minimization procedure in [Xu et al., 2018], as λ governs a trade-off between quantization and closeness to θ. To see that this is a strict generalization, note that for any λ the solution of eq. (12) will be an interpolation between the input θ and its Euclidean projection to $\mathcal{Q}$. As $\lambda \to +\infty$, the prox operator collapses to the projection.

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 quantization, we use an approximate version of the alternating prox operator eq. (12): compute $\widetilde{\theta} = \mathrm{prox}_{\lambda R}(\theta)$ by initializing at $\widetilde{\theta} = \theta$ and repeating

$$\widehat{\theta} = \mathsf{q}(\widetilde{\theta}) \quad \text{and} \quad \widetilde{\theta} = \frac{\theta + 2\lambda\widehat{\theta}}{1+2\lambda}, \tag{13}$$

where q is the ternary quantizer defined as

$$\mathsf{q}(\theta) = \theta^+\mathbf{1}\{\theta \ge \Delta\} + \theta^-\mathbf{1}\{\theta \le -\Delta\},\ \ \Delta = \frac{0.7}{d}\left\|\theta\right\|_1,\ \ \theta^+ = \overline{\theta|_{i:\theta_i\ge\Delta}},\ \ \theta^- = \overline{\theta|_{i:\theta_i\le-\Delta}}. \tag{14}$$

This is a straightforward extension of the TWN quantizer [Li and Liu, 2016] that allows different levels for positives and negatives. We find that two rounds of alternating computation in eq. (13) achieves a good performance, which we use in our experiments.

E.2 Homotopy method for 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.

We let λ_t 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. There is nothing special about the linear increasing scheme, but it is simple enough and works well as we shall see in the experiments.

F Experiments on LSTMs

Problem setup We perform language modeling with LSTMs Hochreiter and Schmidhuber [1997] on the Penn Treebank (PTB) dataset [Marcus et al., 1993], 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 $\{\alpha_1, \ldots, \alpha_k\}$.

Method We compare our multi-bit ProxQuant to the state-of-the-art alternating minimization algorithm with straight-through gradients [Xu et al., 2018]. 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 (PPW, lower is better) in Table 2. The performance of ProxQuant is comparable with the Straight-through gradient method. On Binary LSTMs, ProxQuant-Binary beats BinaryConnect by a large margin. These results demonstrate that ProxQuant offers a powerful alternative 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)	106.2	90.0	87.2	

G Sign change experiment

We experimentally 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:

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

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.

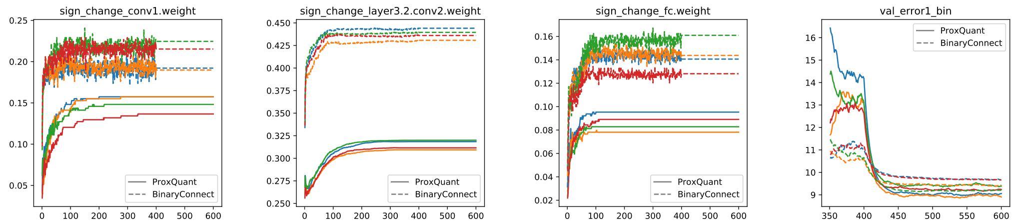
Recall in our CIFAR-10 experiments (Section 3.1), for both BinaryConnect and ProxQuant, we initialize at a good full-precision net θ_0 and stop at a converged binary network $\widehat{\theta} \in \{\pm 1\}^d$. We are interested in $\mathsf{SignChange}(\theta_0, \theta_t)$ along the training path, as 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.

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 in Appendix G.1). 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 than ProxQuant. This result here is also consistent with Theorem 4.1: the signs in BinaryConnect never stop changing until we manually freeze the signs at epoch 400.

G.1 Detailed sign change results on ResNet-20

[2]We thank Xu et al. [2018] 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 $\mathsf{q}_{\rm alt}(\theta)$, they multiply by a constant 0.3 before taking the gradient; in other words, their quantizer is a rescaled alternating quantizer: $\theta \mapsto 0.3\mathsf{q}_{\rm alt}(\theta)$. This scaling step 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 does not involve a scaling step and achieves better PPW than this unscaled ALT straight-through method.



(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 conv layer of size $16 \times 3 \times 3 \times 3$; (b) The last conv 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).

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	BC	9.489 (0.223)	0.383 (0.006)
(8.06)	PQ-B	**9.146** (0.212)	**0.276** (0.020)
FP-Net 2	BC	9.745 (0.422)	0.381 (0.004)
(8.31)	PQ-B	**9.444** (0.067)	**0.288** (0.002)
FP-Net 3	BC	9.383 (0.211)	0.359 (0.001)
(7.73)	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	BC	9.664, 9.430, 9.198, 9.663	0.386, 0.377, 0.390, 0.381
(8.06)	PQ-B	9.058, 8.901, 9.388, 9.237	0.288, 0.247, 0.284, 0.285
FP-Net 2	BC	9.456, 9.530, 9.623, 10.370	0.376, 0.379, 0.382, 0.386
(8.31)	PQ-B	9.522, 9.474, 9.410, 9.370	0.291, 0.287, 0.289, 0.287
FP-Net 3	BC	9.107, 9.558, 9.538, 9.328	0.360, 0.357, 0.359, 0.360
(7.73)	PQ-B	9.284, 8.866, 9.301, 8.884	0.275, 0.276, 0.276, 0.275