Bayesian Optimized 1-Bit CNNs

Jiaxin Gu,[1,†] Junhe Zhao,[1,†] Xiaolong Jiang,[1,2,3] Baochang Zhang,[1,*]
Jianzhuang Liu,[4] Guodong Guo,[2,3] Rongrong Ji[5,6]

[1]Beihang University, Beijing, China
[2]Institute of Deep Learning, Baidu Research, Beijing China
[3]National Engineering Laboratory for Deep Learning Technology and Application
[4]Huawei Noah's Ark Lab, China
[5]School of Information Science and Engineering, Xiamen University, Fujian, China
[6]Peng Cheng Lab, Shenzhen, China
[*]Corresponding author, email: bczhang@buaa.edu.cn
[†] Co-first author

Abstract

Deep convolutional neural networks (DCNNs) have dominated the recent developments in computer vision through making various record-breaking models. However, it is still a great challenge to achieve powerful DCNNs in resource-limited environments, such as on embedded devices and smart phones. Researchers have realized that 1-bit CNNs can be one feasible solution to resolve the issue; however, they are baffled by the inferior performance compared to the full-precision DCNNs. In this paper, we propose a novel approach, called Bayesian optimized 1-bit CNNs (denoted as BONNs), taking the advantage of Bayesian learning, a well-established strategy for hard problems, to significantly improve the performance of extreme 1-bit CNNs. We incorporate the prior distributions of full-precision kernels and features into the Bayesian framework to construct 1-bit CNNs in an end-to-end manner, which have not been considered in any previous related methods. The Bayesian losses are achieved with a theoretical support to optimize the network simultaneously in both continuous and discrete spaces, aggregating different losses jointly to improve the model capacity. Extensive experiments on the ImageNet and CIFAR datasets show that BONNs achieve the best classification performance compared to state-of-the-art 1-bit CNNs.

1. Introduction

Deep convolutional neural networks (DCNNs) have exhibited their superior feature representation power in both
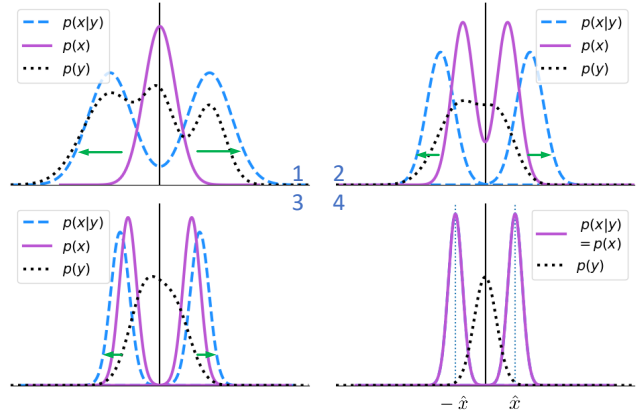

Figure 1. The evolution of the prior $p(\boldsymbol{x})$, the distribution of the observation $\boldsymbol{y}$, and the posterior $p(\boldsymbol{x}|\boldsymbol{y})$ during learning, where $\boldsymbol{x}$ is the latent variable representing the full-precision parameters and $\boldsymbol{y}$ is the quantization error. At the beginning, the parameters $\boldsymbol{x}$ are initialized according to a single-mode Gaussian distribution. When our learning algorithm converges, the ideal case is that (i) $p(\boldsymbol{y})$ becomes a Gaussian distribution $\mathcal{N}(0, \nu)$, which corresponds to the minimum reconstruction error, and (ii) $p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{x})$ is a Gaussian mixture distribution with two modes where the binarized values $\hat{\boldsymbol{x}}$ and $-\hat{\boldsymbol{x}}$ are located.

low-level [5, 13] and high-level vision tasks [8, 14, **?**, **?**]. However, this superiority comes with prohibitive computation and storage overheads. In most cases, heavy parameters of DCNNs are stored as floating point numbers, each of which usually takes 32 bits, and the convolution operation is implemented as matrix multiplication between floating-point operands. These floating-point based operations are time-consuming and storage-demanding. Consequently,

DCNNs are infeasible to be deployed on edge devices such as cellphones and drones, due to the conflict between high demands and limited resources. To tackle this problem, substantial approaches have been explored to compress DCNNs by pruning [11, 7] or quantization [2].

Quantization approximates full-precision values with lower-precision ones, therefore it can simultaneously accelerate the convolution operation and save storage expense. In particular, 1-bit convolution neural networks (1-bit CNNs) are the extreme cases of quantization, whose convolution kernels and activations are binarized, such as ±1 in [3] or $\pm\alpha_l$ in [18]. Recently, DoReFa-Net [24] exploits 1-bit convolution kernels with low bit-width parameters and gradients to accelerate both the training and inference phases. Differently, ABC-Net [12] adopts multiple binary weights and activations to approximate full-precision weights such that the prediction accuracy degradation can be alleviated. Beyond that, modulated convolutional networks are presented in [20] to only binarize the kernels, and achieve better results than the compared baselines. Leng *et al.* [10] borrows the idea from ADMM, which compresses deep models with network weights represented by only a small number of bits. Bi-real net [15] explores a new variant of residual structure to preserve the real activations before the sign function and proposes a tight approximation to the derivative of the non-differentiable sign function. Zhuang *et al.* [25] present $2\sim4$-bit quantization using a two-stage approach to alternately quantize the weights and activations, and provide an optimal tradeoff among memory, efficiency and performance. Furthermore, WAGE [22] is proposed to discretize both the training and inference processes, and it quantizes not only weights and activations, but also gradients and errors. In [6], a quantization method is introduced based on a discrete back propagation algorithm via projection for a better 1-bit CNNs. Other practices are studied in [**?**, **?**, **?**] with improvements over previous works.

Although these prevailing 1-bit CNNs use much less storage than conventional full-precision CNNs, yet compared to full-precision CNNs, they suffer from degraded accuracy in applications. Two reasons should account for this degradation: 1) the relationship between full-precision and 1-bit CNNs is not fully investigated for promoting the performance of 1-bit CNNs; 2) Bayesian learning, as a well-established strategy for global optimization [17, 1], is overlooked in the field of 1-bit CNNs, although it can be beneficial to the optimization of 1-bit CNNs according to our observations.

In this paper, a Bayesian learning algorithm is proposed to optimize our 1-bit CNNs, leading to improved accuracy and efficiency. Theoretically speaking, we achieve two novel Bayesian losses, with the help of Bayesian learning, to solve the difficult problem of CNNs binarization. For 1-bit CNNs, the full-precision kernels are binarized to

two quantization values (centers) gradually. Ideally, the quantization error is minimized when the full-precision kernels follow a Gaussian mixture model with each Gaussian centered at each quantization value. Given two centers for 1-bit CNNs, two Gaussians forming the mixture model are employed to model the full-precision kernels. The whole procedure can be illustrated by Fig. 1, when the learning algorithm converges with a binary quantization, the ideal result should be that: (1) the reconstruction error is minimized, and (2) the distribution of the parameters is a Gaussian mixture model with two modes centered at the binarized values separately. This assumption leads to our two new losses, referred to as the Bayesian kernel loss and Bayesian feature loss. The advantages of these novel losses are twofold. On one hand, they can be jointly applied with the conventional cross-entropy loss within the same back-propagation pipeline, such that the advantages of Bayesian learning is intrinsically inherited to optimize difficult problems. On the other hand, they can comprehensively supervise the training process of 1-bit CNNs with respect to both the kernel distribution and the feature distribution. In summary, the contributions of this paper include:

(1) We propose two novel Bayesian losses to optimize 1-bit CNNs, which are designed via exploiting Bayesian learning to fully investigate the intrinsic relationship between full-precision and 1-bit CNNs in terms of kernels and features.

(2) We develop a novel Bayesian learning algorithm to build 1-bit CNNs in an end-to-end manner. The proposed losses supervise the training process considering both the kernel distribution and the feature distribution, which are more comprehensive and efficient.

(3) Our models achieve the best classification performance compared to other state-of-the-art 1-bit CNNs on the ImageNet and CIFAR datasets.

2. Proposed Method

Bayesian learning is one of the mainstreams in machine learning, which has been applied to building and analyzing neural networks to accomplish computer vision tasks [1, 16]. In the paper, we leverage the efficacy of Bayesian learning to build 1-bit CNNs in an end-to-end manner. In particular, we lead to two novel Bayesian losses, based on which we optimize 1-bit CNNs with improved efficiency and stability. In a unified theoretical framework, these Bayesian losses not only take care of the kernel weight distribution specific in 1-bit CNNs, but also supervise the feature distribution. Fig. 2 shows how the losses interact with a CNN backbone. For clarity, in Table 1 we first describe the main notation used in the following sections.
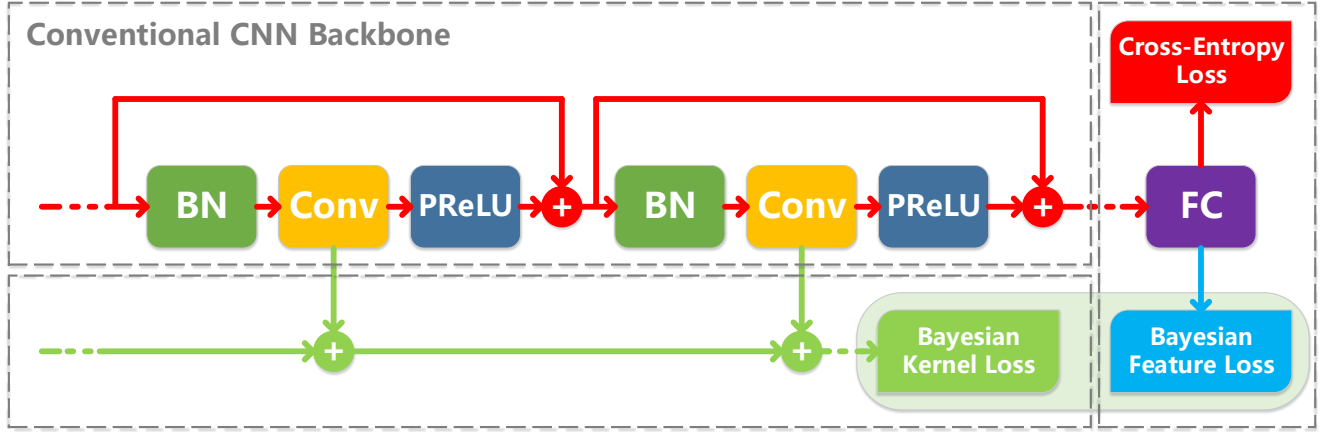
Figure 2. By considering the prior distributions of the kernels and features in the Bayesian framework, we achieve two new Bayesian losses to optimize the calculation of 1-bit CNNs. The Bayesian kernel loss improves the layer-wise kernel distribution of each convolution layer, while the Bayesian feature loss introduces the intra-class compactness to alleviate the disturbance induced by the quantization process. Note that the Bayesian feature loss is only applied to the fully-connected layer.

Table 1. A brief description of the main notation used in the paper.

$\boldsymbol{X}_i^l$: full-precision kernel vector	$\boldsymbol{w}^l$: modulation vector	$\boldsymbol{\mu}_i^l$: mean of $\boldsymbol{X}_i^l$	$\boldsymbol{\Psi}^l$: covariance of $\boldsymbol{X}^l$
$\hat{\boldsymbol{X}}_i^l$: quantized kernel vector	$\boldsymbol{f}_m$: features of class m	λ: trade-off scalar for L_B	$\boldsymbol{c}_m$: mean of $\boldsymbol{f}_m$
i: kernel index	l: layer index	m: class index	k: dimension index
I_l: number of kernels at layer l	L: number of layers	M: number of classes	ν: variance of quantization error

2.1. Bayesian Losses

In state-of-the-art 1-bit CNNs [10, 18, 6], the optimization involves in both continuous and discrete spaces. In particular, training a 1-bit CNN involves three steps: forward pass, backward pass, and parameter update through gradients. The binarized weights ($\hat{\boldsymbol{x}}$) are only considered during the forward pass (inference) and gradient calculation. After updating the parameters, we have the full-precision weights ($\boldsymbol{x}$). As revealed in [10, 18, 6], how to connect $\hat{\boldsymbol{x}}$ with $\boldsymbol{x}$ is the key to determine the network performance. In this paper, we propose to solve it in a probabilistic framework, in order to obtain optimal 1-bit CNNs.

Bayesian kernel loss. We start with the fundamentals: given a parameter, we want it to be as close as possible before and after quantization, such that the quantization effect is minimized. Then, define

$$\boldsymbol{y} = \boldsymbol{w}^{-1} \circ \hat{\boldsymbol{x}} - \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbf{R}^n$ are the full-precision and quantized vectors respectively, $\boldsymbol{w} \in \mathbf{R}^n$ denotes a learned vector to reconstruct $\boldsymbol{x}$, $\circ$ represents the Hadamard product, and $\boldsymbol{y}$ is the reconstruction error assumed to obey a Gaussian prior with zero mean and variance ν. Given $\boldsymbol{y}$, we seek $\hat{\boldsymbol{x}}$ for binary quantization (1-bit CNNs) such that:

$$\hat{\boldsymbol{x}} = \max p(\boldsymbol{x}|\boldsymbol{y}), \tag{2}$$

which indicates that under the most probable $\boldsymbol{y}$ (corresponding to $\boldsymbol{y} = \boldsymbol{0}$ and $\boldsymbol{x} = \boldsymbol{w}^{-1} \circ \hat{\boldsymbol{x}}$, *i.e.*, the minimum reconstruction error), the distribution of the latent variable $\boldsymbol{x}$ is a Gaussian mixture with two modes, locating at the quantization values, as shown in Fig. 1. And we have:

$$
\begin{aligned}
p(\boldsymbol{x}|\boldsymbol{y}) \propto & \exp(-\frac{1}{2}(\boldsymbol{x} - \widetilde{\boldsymbol{\mu}})^T \boldsymbol{\Psi}^{-1}(\boldsymbol{x} - \widetilde{\boldsymbol{\mu}})) \\
& + \exp(-\frac{1}{2}(\boldsymbol{x} + \widetilde{\boldsymbol{\mu}})^T \boldsymbol{\Psi}^{-1}(\mathbf{x} + \widetilde{\boldsymbol{\mu}})),
\end{aligned}
\tag{3}
$$

where we set $\widetilde{\boldsymbol{\mu}} = \boldsymbol{w}^{-1} \circ \hat{\boldsymbol{x}}$. However, Eq. 2 is difficult to solve. From a Bayesian perspective, we resolve this problem via maximum a posteriori (MAP) estimation:

$$
\begin{aligned}
\max p(\boldsymbol{x}|\boldsymbol{y}) &= \max p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \\
&= \min ||\hat{\boldsymbol{x}} - \boldsymbol{w} \circ \boldsymbol{x}||_2^2 - 2\nu \log(p(\boldsymbol{x})),
\end{aligned}
\tag{4}
$$

where

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp(-\frac{1}{2\nu}||\boldsymbol{y}||_2^2) \propto \exp(-\frac{1}{2\nu}||\hat{\boldsymbol{x}} - \boldsymbol{w} \circ \boldsymbol{x}||_2^2). \tag{5}$$

In Eq. 5, we assume that all the components of the quantization error $\boldsymbol{y}$ are i.i.d, thus resulting in such a simplified form. As shown in Fig. 1, for 1-bit CNNs, $\boldsymbol{x}$ is usually quantized to two numbers with the same absolute value. Thus, $p(\boldsymbol{x})$ is

modeled as a Gaussian mixture with two modes:

$$p(\boldsymbol{x}) = \frac{1}{2}(2\pi)^{-\frac{N}{2}} \det(\boldsymbol{\Psi})^{-\frac{1}{2}} \{\exp(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Psi}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2})$$
$$+ \exp(-\frac{(\boldsymbol{x}+\boldsymbol{\mu})^T \boldsymbol{\Psi}^{-1}(\mathbf{x}+\boldsymbol{\mu})}{2})\}$$
$$\approx \frac{1}{2}(2\pi)^{-\frac{N}{2}} \det(\boldsymbol{\Psi})^{-\frac{1}{2}} \{\exp(-\frac{(\boldsymbol{x}_+-\boldsymbol{\mu}_+)^T \boldsymbol{\Psi}_+^{-1}(\boldsymbol{x}_+-\boldsymbol{\mu}_+)}{2})$$
$$+ \exp(-\frac{(\boldsymbol{x}_-+\boldsymbol{\mu}_-)^T \boldsymbol{\Psi}_-^{-1}(\boldsymbol{x}_-+\boldsymbol{\mu}_-)}{2})\} \tag{6}$$

where $\boldsymbol{x}$ is divided into $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ according to the signs of the elements in $\boldsymbol{x}$ and N is the dimension of $\boldsymbol{x}$. Eq. 6 is obtained based on the assumption that the overlap between $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ is neglected. Accordingly, Eq. 4 can be rewritten as:

$$\min||\hat{\boldsymbol{x}} - \boldsymbol{w} \circ \boldsymbol{x}||_2^2 + \nu(\boldsymbol{x}_+ - \boldsymbol{\mu}_+)^T \boldsymbol{\Psi}_+^{-1}(\boldsymbol{x}_+ - \boldsymbol{\mu}_+)$$
$$+ \nu(\boldsymbol{x}_- + \boldsymbol{\mu}_-)^T \boldsymbol{\Psi}_-^{-1}(\boldsymbol{x}_- + \boldsymbol{\mu}_-) + \nu \log(\det(\boldsymbol{\Psi})), \tag{7}$$

where $\boldsymbol{\mu}_-$ and $\boldsymbol{\mu}_+$ are solved independently. $\det(\boldsymbol{\Psi})$ is accordingly set to be the determinant of the matrix $\boldsymbol{\Psi}_-$ or $\boldsymbol{\Psi}_+$. We call Eq. 7 the Bayesian kernel loss.

Bayesian feature loss. This loss is designed to alleviate the disturbance caused by the extreme quantization process in 1-bit CNNs. Considering the intra-class compactness, the features $\boldsymbol{f}_m$ of the mth class supposedly follow a Gaussian distribution with the mean $\boldsymbol{c}_m$ as revealed in the center loss [21]. Similar to the Bayesian kernel loss, we define $\boldsymbol{y}_f^m = \boldsymbol{f}_m - \boldsymbol{c}_m$ and $\boldsymbol{y}_f^m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}_m)$, and have:

$$\min||\boldsymbol{f}_m - \boldsymbol{c}_m||_2^2 + \sum_{k=1}^{K}\left[\sigma_{m,k}^{-2}(f_{m,k}-c_{m,k})^2 + \log(\sigma_{m,k}^2)\right], \tag{8}$$

which is called the Bayesian feature loss. In Eq. 8, $\sigma_{m,k}$, $f_{m,k}$ and $c_{m,k}$ are the kth elements of $\boldsymbol{\sigma}_m$, $\boldsymbol{f}_m$ and $\boldsymbol{c}_m$, respectively.

2.2. Optimized 1-bit CNNs with Bayesian learning

We employ the two Bayesian losses to facilitate the optimization of 1-bit CNNs. We name this method as Bayesian Optimized 1-bit CNNs (BONNs). Now, we can reformulate the two Bayesian losses for 1-bit CNNs as

follows:

$$L_B = \frac{\lambda}{2}\sum_{l=1}^{L}\sum_{i=1}^{I_l}\{||\hat{\boldsymbol{X}}_i^l - \boldsymbol{w}^l \circ \boldsymbol{X}_i^l||_2^2$$
$$+ \nu(\boldsymbol{X}_{i+}^l - \boldsymbol{\mu}_{i+}^l)^T(\boldsymbol{\Psi}_{i+}^l)^{-1}(\boldsymbol{X}_{i+}^l - \boldsymbol{\mu}_{i+}^l)$$
$$+ \nu(\boldsymbol{X}_{i-}^l + \boldsymbol{\mu}_{i-}^l)^T(\boldsymbol{\Psi}_{i-}^l)^{-1}(\boldsymbol{X}_{i-}^l + \boldsymbol{\mu}_{i-}^l)$$
$$+ \nu \log(\det(\boldsymbol{\Psi}^l))\} + \frac{\theta}{2}\sum_{m=1}^{M}\{||\boldsymbol{f}_m - \boldsymbol{c}_m||_2^2$$
$$+ \sum_{k=1}^{K}\left[\sigma_{m,k}^{-2}(f_{m,k} - c_{m,k})^2 + \log(\sigma_{m,k}^2)\right]\}, \tag{9}$$

where $\boldsymbol{X}_i^l, l \in \{1,...,L\}, i \in \{1,...,I_l\}$ is the vectorization of the ith kernel matrix at the lth convolutional layer, $\boldsymbol{w}^l$ is a vector used to modulate $\boldsymbol{X}_i^l$, and $\boldsymbol{\mu}_i^l$ and $\boldsymbol{\Psi}_i^l$ are the mean and covariance of the ith kernel vector at the lth layer, respectively. Furthermore, we assume the parameters in the same kernel are independent, and thus $\boldsymbol{\Psi}_i^l$ become a diagonal matrix with the identical value $(\sigma_i^l)^2$, the variance of the ith kernel of the lth layer. In this case, the calculation of the inverse of $\boldsymbol{\Psi}_i^l$ is speeded up, and also all the elements of $\boldsymbol{\mu}_i^l$ are identical, equal to μ_i^l. Note that in our implementation, all elements of $\boldsymbol{w}^l$ are replaced by their average during the forward process. Accordingly, only a scalar instead of a matrix is involved in the inference, and thus the computation is significantly accelerated.

In BONNs, the cross-entropy loss L_s, the Bayesian kernel loss and the Bayesian feature loss are aggregated together to build the total loss as:

$$L = L_S + L_B. \tag{10}$$

The Bayesian kernel loss constrains the distribution of the convolution kernels to a symmetric Gaussian mixture with two modes, and simultaneously, minimizes the quantization error through the $||\hat{\boldsymbol{X}}_i^l - \boldsymbol{w}^l \circ \boldsymbol{X}_i^l||_2^2$ term. Meanwhile the Bayesian feature loss modifies the distribution of the features to reduce the intra-class variation for better classification.

2.3. Backward Propagation

To minimize Eq. 9, we update $\boldsymbol{X}_i^l$, $\boldsymbol{w}^l$, μ_i^l, σ_i^l, $\boldsymbol{c}_m$ and $\boldsymbol{\sigma}_m$ using the stochastic gradient descent (SGD) algorithm, which is elaborated in the following.

2.3.1 Updating X_i^l

We define $\delta_{\boldsymbol{X}_i^l}$ as the gradient of the full-precision kernel $\boldsymbol{X}_i^l$, and have:

$$\delta_{\boldsymbol{X}_i^l} = \frac{\partial L}{\partial \boldsymbol{X}_i^l} = \frac{\partial L_S}{\partial \boldsymbol{X}_i^l} + \frac{\partial L_B}{\partial \boldsymbol{X}_i^l}. \tag{11}$$

For each term in Eq. 11, we have:

$$\frac{\partial L_S}{\partial \boldsymbol{X}_i^l} = \frac{\partial L_S}{\partial \hat{\boldsymbol{X}}_i^l} \frac{\partial \hat{\boldsymbol{X}}_i^l}{\partial (\boldsymbol{w}^l \circ \boldsymbol{X}_i^l)} \frac{\partial (\boldsymbol{w}^l \circ \boldsymbol{X}_i^l)}{\partial \boldsymbol{X}_i^l}$$
$$= \frac{\partial L_S}{\partial \hat{\boldsymbol{X}}_i^l} \circ \mathbb{1}_{-1 \leq \boldsymbol{w}^l \circ \boldsymbol{X}_i^l \leq 1} \circ \boldsymbol{w}^l, \quad (12)$$

$$\frac{\partial L_B}{\partial \boldsymbol{X}_i^l} = \lambda \{ \boldsymbol{w}^l \circ \left[\boldsymbol{w}^l \circ \boldsymbol{X}_i^l - \hat{\boldsymbol{X}}_i^l \right]$$
$$+ \nu [(\boldsymbol{\sigma}_i^l)^{-2} \circ (\boldsymbol{X}_{i+}^l - \boldsymbol{\mu}_{i+}^l) \quad (13)$$
$$+ (\boldsymbol{\sigma}_i^l)^{-2} \circ (\boldsymbol{X}_{i-}^l + \boldsymbol{\mu}_{i-}^l)],$$

where $\mathbb{1}$ is the indicator function, which is widely used to estimate the gradient of non-differentiable parameters [18], and $(\boldsymbol{\sigma}_i^l)^{-2}$ is a vector whose all elements are equal to $(\sigma_i^l)^{-2}$.

2.3.2 Updating $\boldsymbol{w}^l$

Likewise, $\delta_{\boldsymbol{w}^l}$ is composed of the following two parts:

$$\delta_{\boldsymbol{w}^l} = \frac{\partial L}{\partial \boldsymbol{w}^l} = \frac{\partial L_S}{\partial \boldsymbol{w}^l} + \frac{\partial L_B}{\partial \boldsymbol{w}^l}. \quad (14)$$

For each term in Eq. 14, we have:

$$\frac{\partial L_S}{\partial \boldsymbol{w}^l} = \sum_{i=1}^{I_l} \frac{\partial L_S}{\partial \hat{\boldsymbol{X}}_i^l} \frac{\partial \hat{\boldsymbol{X}}_i^l}{\partial (\boldsymbol{w}^l \circ \boldsymbol{X}_i^l)} \frac{\partial (\boldsymbol{w}^l \circ \boldsymbol{X}_i^l)}{\partial \boldsymbol{w}^l}$$
$$= \sum_{i=1}^{I_l} \frac{\partial L_S}{\partial \hat{\boldsymbol{X}}_i^l} \circ \mathbb{1}_{-1 \leq \boldsymbol{w}^l \circ \boldsymbol{X}_i^l \leq 1} \circ \boldsymbol{X}_i^l, \quad (15)$$

$$\frac{\partial L_B}{\partial \boldsymbol{w}^l} = \lambda \sum_{i=1}^{I_l} (\boldsymbol{w}^l \circ \boldsymbol{X}_i^l - \hat{\boldsymbol{X}}_i^l) \circ \boldsymbol{X}_i^l. \quad (16)$$

2.3.3 Updating μ_i^l and σ_i^l

Note that we use the same μ_i^l and σ_i^l for each kernel, so the gradients here are scalars. The gradients $\delta_{\mu_i^l}$, and $\delta_{\sigma_i^l}$ are computed as:

$$\delta_{\mu_i^l} = \frac{\partial L}{\partial \mu_i^l} = \frac{\partial L_B}{\partial \mu_i^l}$$
$$= \frac{\lambda \nu}{K_{I_l}} \sum_{k=1}^{K_{I_l}} \begin{cases} (\sigma_i^l)^{-2} (\mu_i^l - X_{i,k}^l), & X_{i,k}^l \geq 0, \\ (\sigma_i^l)^{-2} (\mu_i^l + X_{i,k}^l), & X_{i,k}^l < 0, \end{cases} \quad (17)$$

$$\delta_{\sigma_i^l} = \frac{\partial L}{\partial \sigma_i^l} = \frac{\partial L_B}{\partial \sigma_i^l}$$
$$= \frac{\lambda \nu}{K_{I_l}} \sum_{k=1}^{K_{I_l}} \begin{cases} -(\sigma_i^l)^{-3}(X_{i,k}^l - \mu_i^l)^2 + (\sigma_i^l)^{-1}, X_{i,k}^l \geq 0, \\ -(\sigma_i^l)^{-3}(X_{i,k}^l + \mu_i^l)^2 + (\sigma_i^l)^{-1}, X_{i,k}^l < 0, \end{cases} \quad (18)$$

Algorithm 1 Optimized 1-bit CNN with Bayesian learning

Input:
    The training dataset; the full-precision kernels $\boldsymbol{X}$; the modulation vector $\boldsymbol{w}$; the learning rate η, regularization parameter λ, θ and variance ν.

Output:
    The BONN based on the updated $\boldsymbol{X}, \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{c}_m, \boldsymbol{\sigma}_m$.

1: Initialize $\boldsymbol{X}$ and $\boldsymbol{w}$ randomly, and then estimate $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ based on the average and variance of $\boldsymbol{X}$, respectively;
2: **repeat**
3:     // Forward propagation
4:     **for** $l = 1$ to L **do**
5:       $\hat{\boldsymbol{X}}_i^l = \boldsymbol{w}^l \circ \text{sign}(\boldsymbol{X}_i^l), \forall i$; // Each element of $\boldsymbol{w}^l$ is replaced by the average of all elements.
6:       Perform activation binarization; // Using the sign function
7:       Perform 2D convolution with $\hat{\boldsymbol{X}}_i^l, \forall i$;
8:     **end for**
9:     // Backward propagation
10:    Compute $\delta_{\hat{\boldsymbol{X}}_i^l} = \frac{\partial L_S}{\partial \hat{\boldsymbol{X}}_i^l}, \forall l, i$;
11:    **for** $l = L$ to 1 **do**
12:      Calculate $\delta_{\boldsymbol{X}_i^l}, \delta_{\boldsymbol{w}^l}, \delta_{\mu_i^l}, \delta_{\sigma_i^l}$; // using Eqs. 11~18
13:      Update parameters $\boldsymbol{X}_i^l, \boldsymbol{w}^l, \mu_i^l, \sigma_i^l$ using SGD;
14:    **end for**
15:    Update $\boldsymbol{c}_m, \boldsymbol{\sigma}_m$;
16: **until** the algotirhm converges.

---

where $X_{i,k}^l, k \in \{1, ..., K_{I_l}\}$, denotes the kth element of vector $\boldsymbol{X}_i^l$. We update $\boldsymbol{c}_m$ using the same strategy as the center loss [21] in the fine-tuning process, while updating $\sigma_{m,k}$ based on L_B is straightforward, which is not elaborated here for brevity. The above equations show that the proposed method is trainable in an end-to-end manner. Finally we summarize the whole learning procedure in Algorithm 1.

3. Experiments

We perform the image classification task on the CIFAR-10/100 [9] and ILSVRC12 ImageNet datasets [4] to evaluate the performance of BONNs. Considering the favorable generalization capability of our method, BONNs could be integrated in any DCNN variants. For a fair comparison with other state-of-the-art 1-bit CNNs, we apply Wide-Resnet (WRN) [23] and ResNet18 [8] as the full-precision backbone networks. In the following experiments, both the kernels and the activations are binarized. The leading performances reported in the following sections verify the superiority of our BONNs.

3.1. Datasets and Implementation Details

3.1.1 Datasets

CIFAR-10 [9] is a natural image classification dataset, composed of a training set and a test set, each with 50,000 and 10,000 32×32 color images, respectively. These images span across 10 different classes, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships and trucks. Comparatively, CIFAR-100 is a more comprehensive dataset containing 100 classes. On CIFAR-10/100, WRNs are employed as the backbones of BONNs. In comparison, ILSVRC12 ImageNet object classification dataset [4] is more diverse and challenging. It contains 1.2 million training images, and 50,000 validation images, across 1000 classes. For comparing with other state-of-the-art methods on this dataset, we adopt ResNet18 as the backbone to verify the effectiveness and superiority of our BONNs.

3.1.2 WRN

The structure of WRN is similar to ResNet in general. Yet additionally, a depth factor k is introduced to control the feature map depth expansion through 3 stages, while the spatial dimension of the features is kept the same. For brevity, we set k to 1 in the experiments. Besides, the number of channels in the first stage is another important parameter in WRN. We set it to 16 and 64, thus resulting in two network configurations: 16-16-32-64 and 64-64-128-256. In the 64-64-128-256 network, a dropout layer with a ratio of 0.3 is added to prevent overfitting. The learning rate is initially set to 0.01, which decays by 20% per 60 epochs until reaching the maximum epoch of 200 on CIFAR-10/100. We set ν to $1e-4$ for quantization error in WRN. Bayesian feature loss is only used in the fine-tuning process. Other training details are the same as those described in [23]. WRN-22 denotes a network with 22 convolutional layers and similarly for WRN-40.

3.1.3 ResNet18

For ResNet18, we binarize the features and kernels in the backbone convolution layers without convolution layers in shortcuts, following the settings and network modifications in Bi-Real Net [15]. The SGD algorithm is with a momentum of 0.9 and a weight decay of $1e-4$. The learning rate for $\boldsymbol{w}^l, \sigma_i^l$ is set to 0.01, while for $\boldsymbol{X}_i^l, \mu_i^l$ and other parameters the rates are set to 0.1. ν is set to $1e-3$ for the quantization error in ResNet18. The strategy of the learning rate decay is also employed, which is a degradation of 10% for every 30 epochs before the algorithm reaches the maximum epoch of 70.
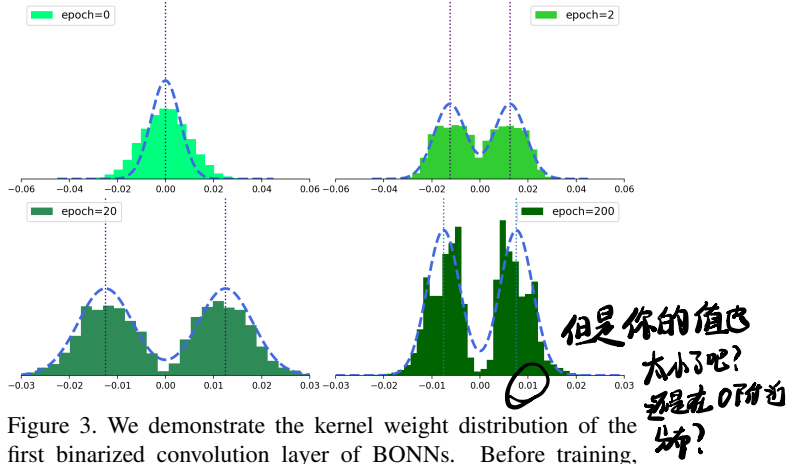

Figure 3. We demonstrate the kernel weight distribution of the first binarized convolution layer of BONNs. Before training, we initialize the kernels as a single-mode Gaussian distribution. From the 2th epoch to the 200th epoch, with λ fixed to $1e-4$, the distribution of the kernel weights becomes more and more compact with two modes, which confirms that the Bayesian kernel loss can regularize the kernels into a promising distribution for binarization.
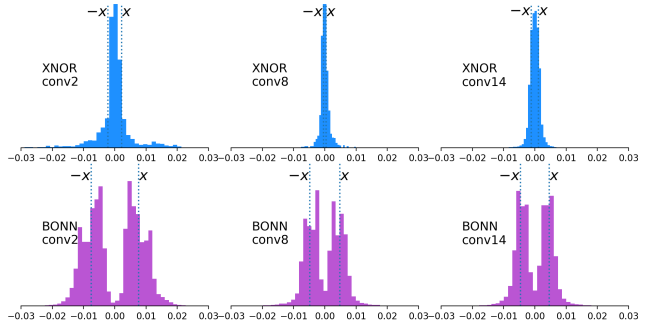

Figure 4. Weight distribution of XNOR and BONN, both based on WRN22 (2nd, 8th and 14th convolutional layers) after 200 epochs. The weight distribution difference between XNOR and BONN indicates that the kernels are regularized with our proposed Bayesian kernel loss, across the convolutional layers.

Table 2. Effect of using or not using the Bayesian losses on the ImageNet dataset. The backbone is ResNet18.

Bayesian kernel loss			✓		✓
Bayesian feature loss				✓	✓
Accuracy	Top-1	56.3	58.3	58.4	**59.3**
	Top-5	79.8	80.8	80.8	**81.6**

3.2. Ablation Study

In this section, we evaluate the effects of the hyper-parameters on the performance of BONNs, including λ and θ. The Bayesian kernel loss and the Bayesian feature loss are balanced by λ and θ, respectively, for adjusting the distributions of kernels and features in a better form. CIFAR-10 and WRN22 are used in the experiments. The

Table 3. Test accuracies on the CIFAR-10/100 datasets. BONNs are based on WRNs [18]. We calculate the number of parameters for each model and the numbers refer to the models on CIFAR-10. Note that for the full-precision models, each parameter takes 32 bits, while for the binary models, each takes only 1 bit.

Model	Kernel stage	#Param	Dataset	
			CIFAR-10	CIFAR-100
WRN22	16-16-32-64	0.27M	91.66	67.51
XNOR-Net	16-16-32-64	0.27M	81.90	53.17
BONN	16-16-32-64	0.27M	**87.34**	**60.91**
WRN22	64-64-128-256	4.33M	94.96	-
XNOR-Net	64-64-128-256	4.33M	88.52	-
BONN	64-64-128-256	4.33M	**92.36**	-
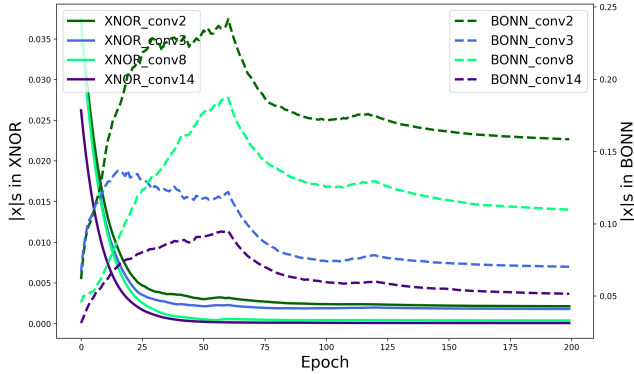
Figure 5. The evolution of the binarized values, $|x|$s, during the training process of XNOR and BONN. They are both based on WRN22 (2nd, 3rd, 8th and 14th convolutional layers) and the curves are not sharing the same y-axis. The binarized values of XNOR Net tend to converge to small and similar values but these of BONN are learned diversely.

implementation details are given below.

We first vary λ and also set it to zero for validating the influence of the Bayesian kernel loss on the kernel distribution. The utilization of the Bayesian kernel loss effectively improves the accuracy on CIFAR-10. But the accuracy does not increase with λ, which indicates what we need is not a larger λ, but a proper λ to reasonably balance the relationship between the cross-entropy loss and the Bayesian kernel loss. For example, when λ is set to $1e-4$, we obtain an optimal balance and the classification accuracy is the best.

The hyper-parameter θ dominates the intra-class variations of the features, and the effect of the Bayesian feature loss on the features is also investigated by changing θ. The results illustrate that the classification accuracy varies in a way similar to λ, which verifies that the Bayesian feature loss can lead to a better classification accuracy when a proper θ is chosen.

To understand the Bayesian losses better, we carry out an experiment to examine how each loss affects the performance. According to the above experiments, we set λ to

$1e-4$ and θ to $1e-3$, if they are used. As shown in Table 2, both the Bayesian kernel loss and the Bayesian feature loss can independently improve the accuracy on ImageNet, and when applied together, the Top-1 accuracy reaches the highest value of 59.3%.

Besides, Fig. 3 illustrates the distribution of the kernel weights, with λ fixed to $1e-4$. During the training process, the distribution is gradually approaching the two-mode GMM as assumed previously, confirming the effectiveness of the Bayesian kernel loss in a more intuitive way. We also compare the kernel weight distribution between XNOR Net and BONN. As is shown in Fig. 4, the kernel weights learned in XNOR Net distribute tightly around the threshold value but these in BONN are regularized in a two-mode GMM style. The Fig. 5 shows the evolution of the binarized values during the training process of XNOR Net and BONN. The two different pattern indicates the binarized values learned in BONN are more diverse.

3.3. Results on the CIFAR-10/100 datasets

We first evaluate our proposed BONNs in comparison with XNOR-Net [18] with WRN backbones and also report the accuracy of full-precision WRNs on CIFAR-10 and CIFAR-100. Three WRN variants are chosen for a comprehensive comparison: 22-layer WRNs with the kernel stage of 16-16-32-64 and 64-64-128-256. We also use data augmentation where each image is with a padding size of 4 and is randomly cropped into 32×32 for CIFAR-10/100. Table 3 indicates that BONNs outperform XNOR-Net on both datasets by a large margin in all the three cases. Compared with the full-precision WRNs, BONNs eliminate the accuracy degradation to an acceptable level, *e.g.*, only 2.6% left on the backbone WRN22 with 64-64-128-256, which verifies the advantage of our method in building 1-bit CNNs.

3.4. Results on the ImageNet dataset

To further evaluate the performance of our method, we evaluate BONNs on the ImageNet dataset. Fig. 6 shows the curves of the Top-1 and Top-5 training/test accuracies.
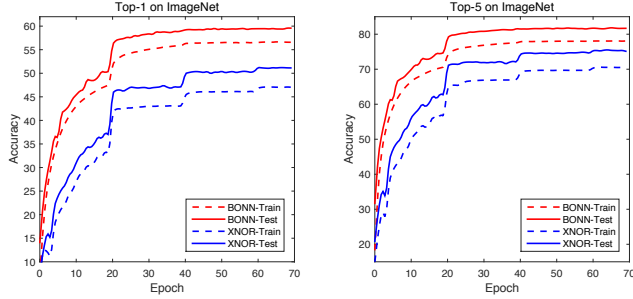
Figure 6. Training and Test accuracies on ImageNet when $\lambda = 1e-4$, which shows the superiority of the proposed BONN over XNOR-Net. The backbone of the two networks is ResNet18.

Table 4. Test accuracies on ImageNet. 'W' and 'A' refer to the weight and activation bitwidth respectively. The backbone of all the models is ResNet18.

Model	W	A	Top-1	Top-5
ResNet18	32	32	69.3	89.2
BWN	1	32	60.8	83.0
DoReFa-Net	1	4	59.2	81.5
TBN	1	2	55.6	79.0
BNN	1	1	42.2	67.1
XNOR-Net	1	1	51.2	73.2
ABC-Net	1	1	42.7	67.6
Bi-Real Net	1	1	56.4	79.5
PCNN	1	1	57.3	80.0
BONN	1	1	**59.3**	**81.6**

Notably, we adopt two data augmentation methods in the training set: 1) cropping the image to the size of 224×224 at a random location, and 2) flipping the image horizontally. In the test set, we simply crop the image to 224×224 in the center. ResNet18 is the backbone, only with slight structure adjustments as described in [15].

We compare the performance of BONN with other state-of-the-art quantized networks, including BWN [18], DoReFa-Net [24], TBN [19], XNOR-Net [18], ABC-Net [12], BNN [3], Bi-Real Net [15] and PCNN [6]. Table 4 indicates that our BONN obtains the highest accuracy among these 1-bit CNNs, in which Bi-Real Net and PCNN perform most similar to BONN, yet BONN outperforms them by about 3% and 2% in Top-1 accuracy, respectively. Moreover, due to the application of the clip function [15], Bi-Real Net is trained in a two-stage procedure which requires an extra cost. It is also worth mentioning that DoReFa-Net and TBN use more than 1-bit to quantize the activations, yet we still get better performance in comparison. These results show that BONNs are not limited to small datasets, but also work well on large datasets. This further verifies the generalization capability of our BONNs.

3.5. Memory Usage and Efficiency Analysis ①第一层stem是fp的 ② 1×1 conv是fp的 ③ fc层是fp的.

In a full-precision network, each parameter requires 32 bits to save it. While in 1-bit CNNs, each parameter is stored with just 1 bit. In BONNs, we follow a strategy adopted by XNOR-Net, which keeps full-precision the parameters in the first convolution layer, all 1×1 convolutions and the fully connected layer. This leads to an overall compression rate of 11.10 for ResNet18. For efficiency analysis, if all of the operands of the convolutions are binary, then the convolutions can be estimated by XNOR and bit-counting operations.[3] In this way, we can get $58\times$ faster on CPUs [18].

4. Conclusion and future work

In this paper, we have proposed Bayesian optimized 1-bit CNNs (BONNs), which take the full-precision kernel and feature distributions into consideration, resulting in a unified Bayesian framework with two new Bayesian losses. The Bayesian losses are used to adjust the distributions of kernels and features towards an optimal solution. Comprehensive studies on the hyper-parameters for the Bayesian losses have been conducted. Extensive experiments on CIFAR and ImageNet demonstrate that BONNs achieve the best classification performance for WRNs and ResNet18, and have superior performance over other 1-bit CNNs. In the future, we plan to explore our proposed BONNs on deeper networks like ResNet34 and on different tasks other than classification.

5. Acknowledgement

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[2] 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.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or- 1. *arXiv preprint arXiv:1602.02830*, 2016.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of the European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[6] J. Gu, C. Li, B. Zhang, J. Han, X. Cao, J. Liu, and D. Doermann. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *AAAI Conference on Artificial Intelligence*, 2019.

[7] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[10] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *AAAI Conference on Artificial Intelligence*, 2018.

[11] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[12] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[13] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision*, pages 85–100, 2018.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[15] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision*, pages 747–763. Springer, 2018.

[16] D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[17] J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

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

[19] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *Proceedings of the European Conference on Computer Vision*, September 2018.

[20] X. Wang, B. Zhang, C. Li, R. Ji, J. Han, X. Cao, and J. Liu. Modulated convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–848, 2018.

[21] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *Proceedings of the European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[22] S. Wu, G. Li, F. Chen, and L. Shi. Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*, 2018.

[23] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*. British Machine Vision Association, 2016.

[24] 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.

[25] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Towards effective low-bitwidth convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.