Training Binary Neural Networks through Learning with Noisy Supervision

Kai Han [1 2] Yunhe Wang [2] Yixing Xu [2] Chunjing Xu [2] Enhua Wu [1 3] Chang Xu [4]

Abstract

This paper formalizes the binarization operations over neural networks from a learning perspective. In contrast to classical hand crafted rules (*e.g.* hard thresholding) to binarize full-precision neurons, we propose to learn a mapping from full-precision neurons to the target binary ones. Each individual weight entry will not be binarized independently. Instead, they are taken as a whole to accomplish the binarization, just as they work together in generating convolution features. To help the training of the binarization mapping, the full-precision neurons after taking sign operations is regarded as some auxiliary supervision signal, which is noisy but still has valuable guidance. An unbiased estimator is therefore introduced to mitigate the influence of the supervision noise. Experimental results on benchmark datasets indicate that the proposed binarization technique attains consistent improvements over baselines.

1. Introduction

Deep convolutional neural networks (CNNs) have achieved much success in many real-world applications such as image recognition (He et al., 2016; Han et al., 2018c), object detection (Ren et al., 2015), and semantic segmentation (Chen et al., 2016). These CNN models usually consume high computational resource, and thus they cannot be easily deployed on embedded devices. A series of model compression and acceleration methods (Han et al., 2016; Chen et al., 2020) have been proposed to reduce the number of parameters and FLOPs of CNNs, including network pruning (Han et al., 2016; Li et al., 2017; Shu et al., 2019), tensor decomposition (Denton et al., 2014), knowledge distillation (Hinton et al., 2015; Chen et al., 2019), efficient model design (Howard et al., 2017; Han et al., 2020), and model quantization (Gupta et al., 2015; Hubara et al., 2016). These methods have significantly promoted the development of deep learning towards real-world mobile applications.

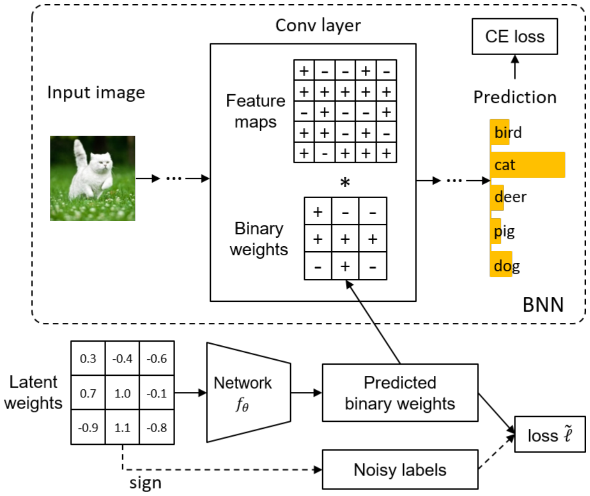


Figure 1. Framework of learning binary neurons with noisy supervision. A network f is utilized to predict the binary weights with supervision of the noisy labels obtained by sign function.

Binary neural networks (BNNs) (Hubara et al., 2016; Rastegari et al., 2016; Lin et al., 2017; Liu et al., 2018; Shen et al., 2020) push neural network quantization to the extreme. 1-bit weights and activations in BNNs can dramatically save computational cost for better real-time inference. BinaryNet (Hubara et al., 2016) first proposed binary networks with both 1-bit weights and activations. XNOR-Net (Rastegari et al., 2016) further improved BinaryNet by introducing channel-wise scale factor of weights and activation. Dorefa-Net (Zhou et al., 2016) utilized layer-wise scale factor to achieve XNOR-Net like performance. ABC-Net (Lin et al., 2017) enhanced the performance by using more weight bases and activation bases, but computation complexity is increased at the same time. These approaches have constantly boosted the performance of BNNs. For example, binary AlexNet in XNOR-Net (Rastegari et al., 2016) achieves a 44.2% top-1 accuracy on the ImageNet classification task, while reducing the convolution parameters for nearly 32× over the full-precision model (56.6% top-1 accuracy).

In constructing the binary neural network, most of existing

[1] State Key Lab of Computer Science, Institute of Software, CAS & University of Chinese Academy of Sciences [2] Noah's Ark Lab, Huawei Technologies [3] University of Macau [4] School of Computer Science, Faculty of Engineering, University of Sydney. Correspondence to: Yunhe Wang <yunhe.wang@huawei.com>, Enhua Wu <ehwu@umac.mo>, Chang Xu <c.xu@sydney.edu.au>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

approaches employ hard thresholding (*e.g.* sign function) to quantize the weights and process each element independently. "Straight through estimator" (STE) (Bengio et al., 2013) is applied to calculate the gradient of sign function. However, the performance of the existing BNNs is far worse than that of the full-precision counterparts, *e.g.* up to 11.6% accuracy drop of binary AlexNet in XNOR-Net. Simply binarizing each individual weight independently does not fully explore the relationship between neurons and may not bring in the optimal solution. Moreover, estimated gradients by STE often lead to inaccurate weights that contain the noise, *i.e.* some of the binary weights have been incorrectly flipped into the opposite values.

In this paper, we learn to binarize neurons with noisy supervision, as shown in Fig. 1. In contrast to classical hand crafted rules to binarize the weights, we suggest a mapping from full-precision neurons to the binary ones. This mapping function can be approximated with a neural network that treats full-precision weights in a filter as a whole for the input. To help the learning of the mapping function, we take the pretrained binary weights as noisy supervisions that are close to the ideal binary neurons. An unbiased estimator is introduced for learning with noisy supervisions and avoiding noise disturbance. In an end-to-end fine-tuning, the proposed method can be a nice alternative of the sign function in BNNs by mining the relationship between neurons and taking advantage of noisy supervisions. Theoretical analysis suggests that the introduced unbiased estimator can converge to the optimal solution of binary weights under the clean distribution. Experiments on benchmark datasets including CIFAR-10 and ImageNet demonstrate that BNNs established using the proposed Learning with Noisy Supervision (LNS) method achieve state-of-the-art performance.

2. Related Work

In this section, we give a brief review of the related work in the field of binary neural networks and learning with noisy labels.

2.1. Binary Neural Networks

Binary neural networks with extremely low memory and computation cost appeal great interest from the community. Binaryconnect (Courbariaux et al., 2015) was proposed to train deep neural networks with binary weights. BinaryNet (Hubara et al., 2016) further quantize both the weights and the activations to 1-bit values, starting the research on pure binary neural networks. XNOR-Net (Rastegari et al., 2016) introduce channel-wise scale factor to improve performance. Dorefa-Net (Zhou et al., 2016) simplifies to layer-wise scale factor and achieve similar performance with XNOR-Net. ABC-Net (Lin et al., 2017) proposes to enhance the performance by using more weight bases and activation bases, but it admittedly needs more memory and computation cost than BinaryNet. There are also several works designing blocks or architectures for binary networks (Liu et al., 2018; Shen et al., 2019). Bireal-Net (Liu et al., 2018) introduces layer-wise identity short-cut, and AutoBNN (Shen et al., 2019) widen or squeeze the channels in an automatic manner. All these binary models utilize STE (Bengio et al., 2013) for gradient back-propagation which would introduce inaccurate gradients for model optimization.

Some works propose new gradient calculation approach instead of STE. Bireal-Net (Liu et al., 2018) and DSQ (Gong et al., 2019) use specially designed activation function for back-propagation. PCNN (Gu et al., 2019) proposes a new discrete back-propagation via projection algorithm to build BNNs. However, most of existing methods quantize each weight independently and ignore their internal relationship.

2.2. Learning with Noisy Labels

To learn more accurate predictions and correct biased information from noisy labels, a number of methods are proposed for learning with noisy labels, which can be divided into three categories:

Label correction aims to correct the wrong labels in the raw labels. The existing methods usually utilize a clean label inference module to correct the noisy labels to the true ones. The inference module can be modeled by neural networks (Lee et al., 2018), graphical models (Xiao et al., 2015), or conditional random fields (Vahdat, 2017). However, the extra clean data or expensive noise detection process is required in these methods, which is unpractical in real-world applications.

Refined training strategies introduce new learning framework for robustness to noisy labels (Jiang et al., 2018; Han et al., 2018b; Yu et al., 2019; Wang et al., 2018; Tanaka et al., 2018). These methods such as MentorNet (Jiang et al., 2018) and Co-teaching (Han et al., 2018b; Yu et al., 2019), change the standard learning process with complex interventions which usually need much effort to adapt and tune.

Loss correction methods improve the standard loss function to suit for noisy labels. One common approach is modeling the noise transition matrix which defines the probability of one class flipped to another one (Natarajan et al., 2013). Backward and Forward (Patrini et al., 2017) introduce two alternative procedures for loss correction, provided knowing the noise transition matrix. A linear layer is added on top of the neural networks for noisy prediction correction in (Goldberger & Ben-Reuven, 2017). Masking (Han et al., 2018a) derive a structure-aware probabilistic model to incorporate the structure prior. Noise robust loss function is another technique dealing with noisy labels, such as generalized cross entropy (Zhang & Sabuncu, 2018), label smoothing regularization (Pereyra et al., 2017), and symmetric cross entropy (Wang et al., 2019).

Binary weights can also be recognized as prediction of a binary classifier, and the biased weights derived from the sign function are exactly the noisy label. Therefore, we present to develop a mapping that can correct the noisy binary weights and obtain BNNs with better performance.

3. Approach

In this section we detail the formulation of our method, including binary weight mapping model and an unbiased estimator with noisy supervision.

3.1. Binary Weight Mapping

In existing binary neural networks such as BinaryNet (Hubara et al., 2016), Bireal-Net (Liu et al., 2018) and Dorefa-Net (Zhou et al., 2016), the weights are usually quantized with the sign function and a scale factor. In particular, the weights before quantization in a convolutional filter are denoted as $W \in \mathbb{R}^{c\times k\times k}$, where c is the number of input channels, $k \times k$ is the kernel size. For simplicity in the following, we omit the scale factor, and quantized binary weights $\tilde{Q} \in \{+1, -1\}^{c\times k\times k}$ can be obtained by

$$\tilde{Q} = \text{sign}(W), \tag{1}$$

where $\text{sign}(\cdot)$ outputs $+1$ for positive input and -1 for negative input. The feature map $X \in \mathbb{R}^{n\times c\times h\times w}$ before convolution are also quantized in a similar way as the weights: $B = \text{sign}(X)$, where $B \in \{+1, -1\}^{n\times c\times h\times w}$, n is the number of samples, and h and w are the height and weight of feature map, respectively. With the binarized weights and feature maps, the convolution computation only involves binary operations, *i.e.* AND and POPCOUNT:

$$Y = B \circledast \tilde{Q}, \tag{2}$$

where $\circledast$ represents convolution operation with binary operations. During training, the back-propagation process of the quantization follows the straight through estimator (Bengio et al., 2013):

$$\frac{\partial \ell_{cls}}{\partial W} \approx \text{clip}\left(\frac{\partial \ell_{cls}}{\partial \tilde{Q}}, -1, 1\right), \tag{3}$$

where ℓ_{cls} is the cross entropy loss function if the neural network is for image classification, and W are the latent full-precision weights to be optimized in iterations. After training, the quantized weights $\tilde{Q} = \text{sign}(W)$ will be kept for the inference.

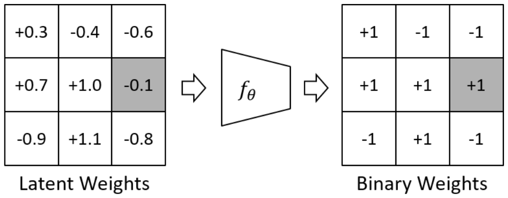


Figure 2. Binary weight mapping. Different from simple sign function, the binarized weight here may be different from the sign of the latent weight in our method (the element in gray is an example).

The simple sign function to binarize the weights cannot take the relationship between elements into consideration and may not be the optimal. In fact, the ideal process to transform the full-precision weights to binary could be complicated and unknown. Instead of trying to fit the handcrafted binarization rules, we propose to binarize neurons through a learned mapping function as shown in Fig. 2. The full-precision weights are taken as a whole, and thus their internal relation can be fully explored and exploited by the mapping model to accomplish the binarization. Formally, the binarization process can be written as

$$\hat{Q} = f_\theta(W), \tag{4}$$

where f_θ is the mapping model with training parameters θ. Compared to sign function, the binarization function in Eq. 4 is learnable and more flexible, which can approximate the binarization for the need to quantize the weights. Sign function only operates on each individual element independently, while the mapping function approximated by a neural network (Eq. 4) can quantize each element by considering its connections with other elements.

We can embed the binary neuron mapping model in every convolutional layer which is needed to be quantized and train the entire neural network end-to-end. However, there is only the final loss (*e.g.* ℓ_{cls}) for supervising the entire network. The mapping model in each layer may be hard to optimize for lack of direct supervision.

3.2. Learning with Noisy Supervision

If the ground-truth binary weights Q are provided, we can force the mapping model to learn the target under the supervision:

$$\ell(\hat{Q}, Q) = \|\hat{Q} - Q\|_F^2, \tag{5}$$

where $\|\cdot\|_F$ is the Frobenius norm of a tensor. Denoting $\hat{q}$ as each element in the predictions $\hat{Q}$, and q as the corresponding ground-truth label in Q, Eq. 5 can be represented as the following for simplicity:

$$\ell(\hat{Q}, Q) = \|\hat{Q} - Q\|_F^2 = \sum(\hat{q} - q)^2. \tag{6}$$

The loss in Eq. 5 is simple and easy to optimize. However, the ground-truth Q is hard to obtain in practice.

If we pretrain a binary neural network as normal (Zhou et al., 2016), we can easily obtain the pretrained binary model with latent weights W and the corresponding binarized weights $\tilde{Q}$ in each layer. Since the gradients to W in Eq. 3 are estimated and inaccurate, the weights W and the quantized $\tilde{Q}$ after optimization are also inaccurate and contain noise, that is, some of the weights have been incorrectly flipped into the opposite values. Nevertheless, as the noise corrupted Q, $\tilde{Q}$ still has valuable guidance to provide auxiliary supervision for learning the mapping model. Eq. 5 can therefore be reformulated as

$$\ell(\hat{Q}, \tilde{Q}) = \|\hat{Q} - \tilde{Q}\|_F^2 = \sum (\hat{q} - \tilde{q})^2, \tag{7}$$

where $\tilde{q}$ is the noisy label for the mapping model. Compared to Eq. 5, the target in Eq. 7 is changed to the noisy binary weights. The mapping function f_θ learned by minimzing the loss in Eq. 7 could be seriously influenced by the noise supervision and may be harmful to the binary neural networks.

To make the full use of the noisy supervision while mitigating the influence of the noise, we seek the solution to learn from the noisy label. Inspired by the developments in noisy label learning (Natarajan et al., 2013), we introduce a loss correction approach to avoid the noise disturbance in the mapping function learning. We view the pretrained weights $\tilde{Q}$ in BNNs as the noisy labels which is the corrupted version of Q. We assume that the noisy labels follow the class-conditional random noise model,

$$P(\tilde{q} = -1 | q = +1) = \rho_{+1}, \tag{8}$$

$$P(\tilde{q} = +1 | q = -1) = \rho_{-1}, \tag{9}$$

where ρ_{+1} is the probability that the negative weight is flipped into $+1$, ρ_{-1} is the probability that the positive weight is flipped into -1, and $\rho_{+1} + \rho_{-1} < 1$. The noise rates ρ_{+1} and ρ_{-1} are two hyper-parameters. In binary neural networks, the number of positive weights and negative weights are similar, so the noise rate ρ_{+1} and ρ_{-1} should be similar as well, *i.e.* $\rho = \rho_{+1} = \rho_{-1}$.

We aim to amend the loss function in Eq. 7 to $\tilde{\ell}(\hat{Q}, \tilde{Q})$ so that we have

$$\mathbb{E}\left[\tilde{\ell}(\hat{Q}, \tilde{Q})\right] = \ell(\hat{Q}, Q), \tag{10}$$

that is $\mathbb{E}\left[\tilde{\ell}(\hat{q}, \tilde{q})\right] = \ell(\hat{q}, q)$. Considering the cases $q = +1$ and $q = -1$ separately, we have the following equations

$$(1 - \rho_{+1})\tilde{\ell}(\hat{q}, +1) + \rho_{+1}\tilde{\ell}(\hat{q}, -1) = \ell(\hat{q}, +1), \tag{11}$$

and

$$(1 - \rho_{-1})\tilde{\ell}(\hat{q}, -1) + \rho_{-1}\tilde{\ell}(\hat{q}, +1) = \ell(\hat{q}, -1). \tag{12}$$

Solving these two equations for $\tilde{\ell}(\hat{q}, +1)$ and $\tilde{\ell}(\hat{q}, -1)$ gives

$$\tilde{\ell}(\hat{q}, +1) = \frac{(1 - \rho_{-1}) l(\hat{q}, +1) - \rho_{+1} l(\hat{q}, -1)}{1 - \rho_{+1} - \rho_{-1}}, \tag{13}$$

and

$$\tilde{\ell}(\hat{q}, -1) = \frac{(1 - \rho_{+1}) l(\hat{q}, -1) - \rho_{-1} l(\hat{q}, +1)}{1 - \rho_{+1} - \rho_{-1}}. \tag{14}$$

By introducing $\tilde{q} \in \{+1, -1\}$, Eqs. 13 and 14 can be merged into a unified loss function

$$\tilde{\ell}(\hat{q}, \tilde{q}) = \frac{(1 - \rho_{-\tilde{q}})\ell(\hat{q}, \tilde{q}) - \rho_{\tilde{q}}\ell(\hat{q}, -\tilde{q})}{1 - \rho_{+1} - \rho_{-1}}. \tag{15}$$

Viewing each element $\tilde{q}$ as one noisy sample, we can learn a mapping model in the presence of noisy label by minimizing the sample average

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}_{\tilde{\ell}}(f) = \frac{1}{|\hat{Q}|} \sum \tilde{\ell}(\hat{q}, \tilde{q}), \tag{16}$$

where $\mathcal{F}$ can be general function class, *e.g.* the neural networks, and $\hat{R}_{\tilde{\ell}}$ stands for the empirical $\tilde{\ell}$-risk on the observed samples. For any fixed $f \in \mathcal{F}$, the above sample average risk can converge to the ℓ-risk under the clean distribution D: $R_{\ell,D}(f)$ even that the predictor is learned with noisy labels whereas the ℓ-risk is computed using true labels as stated in Eq. 10. Theoretically, the performance bound for $\hat{f}$ with respect to the clean distribution D is shown in the following Theorem 1.

Theorem 1. *(Natarajan et al., 2013) With probability at least* $1 - \delta$,

$$R_{\ell,D}(\hat{f}) \leq \min_{f \in \mathcal{F}} R_{\ell,D}(f) + 4L_\rho \mathfrak{R}(\mathcal{F}) + 2\sqrt{\frac{\log(1/\delta)}{2n}}.$$

where $\mathfrak{R}(\mathcal{F}) := \mathbb{E}_{X_i, \epsilon_i}[\sup_{f \in \mathcal{F}} \frac{1}{n}\epsilon_i f(X_i)]$ *is the Rademacher complexity of the function class* $\mathcal{F}$ *and* $L_\rho \leq 2L/(1 - \rho_{+1} - \rho_{-1})$ *is the Lipschitz constant of the loss function* $\tilde{\ell}$. *Note that* ϵ_i*'s are iid Rademacher random variables.*

The unbiased auxiliary loss Eq. 15 is therefore helpful to learn a BNN with binary neural mapping. For the i-th quantized layer, the auxiliary loss $\tilde{\ell}_i$ is investigated:

$$\tilde{\ell}_i = \tilde{\ell}(\hat{Q}, \tilde{Q}) = \sum \tilde{\ell}(\hat{q}, \tilde{q}). \tag{17}$$

The introduced auxiliary loss over the predicted binary neurons from the latent full-precision weights is differential. Both the latent weights W and the mapping parameters θ need to be optimized. The gradient of $\tilde{\ell}_i$ with respect to $\hat{q}$ is easy to obtain:

$$\begin{aligned}
\frac{\partial \tilde{\ell}_i}{\partial \hat{q}} &= \frac{(1 - \rho_{-\tilde{q}})\frac{\partial \ell(\hat{q}, \tilde{q})}{\partial \hat{q}} - \rho_{\tilde{q}}\frac{\partial \ell(\hat{q}, -\tilde{q})}{\partial \hat{q}}}{1 - \rho_{+1} - \rho_{-1}} \\
&= \frac{2(1 - \rho_{-\tilde{q}})(\hat{q} - \tilde{q}) - 2\rho_{\tilde{q}}(\hat{q} + \tilde{q})}{1 - \rho_{+1} - \rho_{-1}} \\
&= 2(\hat{q} - \tilde{q}) - \frac{4\rho_{\tilde{q}}\tilde{q}}{1 - \rho_{+1} - \rho_{-1}}.
\end{aligned} \tag{18}$$

Algorithm 1 Feed-Forward and Back-Propagation Process of Binary Neuron Mapping with Noisy Supervision.

```
Require: Pretrained latent weights $W$, input feature map $B$, the weights of mapping model $\theta$, the loss function $\mathcal{L}$, the learning rate $\eta$.
Feed Foward:
if in training stage then
    Obtain the noisy labels by sign function: $\tilde{Q} = \text{Sign}(W)$ (Eq.1);
    Compute the predicted binary weights via the mapping model: $\hat{Q} = f(W)$ (Eq. 4);
    Compute the auxiliary loss with noisy supervision: $\tilde{\ell}(\hat{Q}, Q) = \sum \tilde{\ell}(\hat{q}, \tilde{q})$ (Eq. 17).
end if
Perform binary convolution: $Y = B \circledast \hat{Q}$.
Backward Propagation:
Compute the gradient of $B$: $\frac{\partial \mathcal{L}}{\partial B} = \frac{\partial \ell_{cls}}{\partial B}$, where $\frac{\partial \ell_{cls}}{\partial B}$ can be calculated as normal neural networks;
Compute the gradient of $W$: $\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \ell_{cls}}{\partial W} + \frac{\partial \ell_i}{\partial W}$, where $\frac{\partial \ell_i}{\partial W}$ is given by Eq. 20 and $\frac{\partial \ell_{cls}}{\partial W}$ can be calculated as normal neural networks;
Compute the gradient of $\theta$: $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \ell_{cls}}{\partial \theta} + \frac{\partial \ell_i}{\partial \theta}$ where $\frac{\partial \ell_i}{\partial \theta}$ is given by Eq. 19 and $\frac{\partial \ell_{cls}}{\partial \theta}$ can be calculated as normal neural networks;
Parameter Update:
Update the latent weights $W \leftarrow W - \eta \frac{\partial \ell}{\partial W}$, and the weights of mapping model $\theta \leftarrow \theta - \eta \frac{\partial \ell}{\partial \theta}$;
```

Then the gradient to θ is given by

$$\frac{\partial \tilde{\ell}_i}{\partial \theta} = \sum_{\hat{q}} \frac{\partial \tilde{\ell}_i}{\partial \hat{q}} \frac{\partial \hat{q}}{\partial \theta}, \tag{19}$$

where $\frac{\partial \hat{q}}{\partial \theta}$ can be calculated with standard chain rule as $f_\theta(\cdot)$ is a neural network. The latent weights W are the input to the mapping model $f_\theta(\cdot)$ to output the binary weight predictions $\hat{q}$, so the gradients are

$$\frac{\partial \tilde{\ell}_i}{\partial W} = \sum_{\hat{q}} \frac{\partial \tilde{\ell}_i}{\partial \hat{q}} \frac{\partial \hat{q}}{\partial W}. \tag{20}$$

where $\frac{\partial \hat{q}}{\partial W}$ can be calculated with back-propagation in the neural network. With the gradients of W and θ, the mapping model with noisy neuron correction loss $\tilde{\ell}_i$ is differential and can be optimized in an end-to-end manner.

For a binary neural network with original classification loss ℓ_{cls}, the overall object function of our method is

$$\mathcal{L} = \ell_{cls} + \alpha \sum_i \tilde{\ell}_i, \tag{21}$$

where α is the trade-off hyper-parameter. The proposed LNS method can be embedded into the training process of binary neural networks and trained in the end-to-end manner. The forward and back-propagation process of our method are listed in Algorithm 1. After training, we obtain the optimized latent weights W and mapping parameters θ. We transform the latent full-precision weights to binary weights $\hat{Q}$ using the mapping neural network, and only keep those binary weights for the inference.

4. Experiments

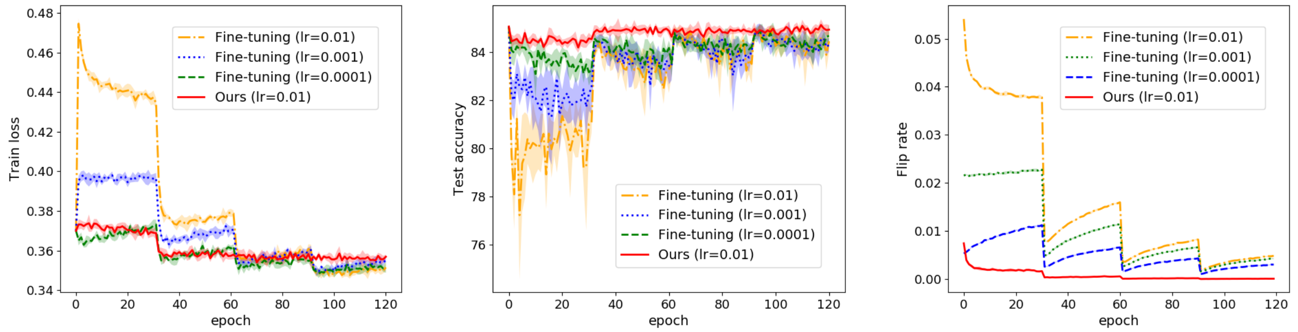
In this section, we evaluate the proposed method on two image classification datasets: CIFAR-10 (Krizhevsky & Hinton, 2009) and ImageNet (ILSVRC12) (Deng et al., 2009), and compare our method with other BNNs.

4.1. Datasets and Experimental Setting

CIFAR-10 CIFAR-10 dataset (Krizhevsky & Hinton, 2009) consists of 60,000 32×32 color images belonging to 10 categories, with 6,000 images per category. There are 50,000 training images and 10,000 test images. For hyper-parameter tuning, 10,000 training images are randomly sampled for validation and the rest images are for training. Data augmentation strategy includes random crop and random flipping as in (He et al., 2016) during training. For testing, we evaluate the single view of the original image for fair comparison.

ImageNet ImageNet ILSVRC 2012 (Deng et al., 2009) is a large-scale image classification dataset which contains over 1.2 million high-resolution natural images for training and 50k validation images in 1,000 classes. The commonly used data augmentation strategy including random crop and flipping in PyTorch examples (Paszke et al., 2019) is adopted for training. We report the single-crop evaluation result using 224×224 center crop from images.

Implementation Details All the models are implemented using PyTorch (Paszke et al., 2019) and conducted on NVIDIA Tesla V100 GPUs. For CIFAR-10, ResNet-20 is used as baseline model. The binary baseline models are trained for 400 epochs with a batch size of 128 and an initial learning rate 0.1. We use the SGD optimizer with the momentum of 0.9 and set the weight decay to 0. Our method is fine-tuned based on the pretrained baseline for 120 epochs using SGD optimizer. The learning rate starts from 0.01 and decayed by 0.1 every 30 epochs. For ImageNet, AlexNet and ResNet-18 are adopted for evaluation. We train the binary baseline models for 120 epochs with a batch size of 256. SGD optimizer is applied with the momentum of 0.9 and the weight decay of 0. The learning rate is set as 0.1 initially and is multiplied by 0.1 at the $70th$, $90th$ and $110th$ epoch, respectively. Our method is fine-tuned from the pretrained baseline for 45 epochs with the initial learning rate



(a) Cross entropy loss *w.r.t.* epoch on CIFAR-10 train set. The mean and std values are plotted.

(b) Accuracy *w.r.t.* epoch on CIFAR-10 test set. The mean and std values are plotted.

(c) Binary weight flip rate *w.r.t.* epoch during training. The mean and std values are plotted (where std values are very small).

Figure 3. Comparison of our method and simple fine-tuning (Ours with lr=0.001/0.0001 are not plotted for tidy figure as they are similar).

0.01 which is decayed by 0.1 every 15 epochs.

In each layer, there is a neural network for binary weight mapping. We simply use a three-layer CNN with weight shape of $2c \times c \times 3 \times 3$, $2c \times 2c \times 3 \times 3$ and $c \times 2c \times 3 \times 3$, respectively. We set padding as 1 and stride as 1 in every layer of the mapping model to keep the size of output unchanged. Batch normalization and ReLU activation are inserted after the intermediate convolutional layers. The mapping model is updated for several epochs for warm start meanwhile the other weights are fixed before fine-tuning.

4.2. Experiments on CIFAR-10

We first conduct detailed studies on CIFAR-10 dataset for the proposed method. The widely used ResNet-20 architecture is adopted as the basic architecture, and Dorefa-Net is used as the baseline quantization method. Following the common setting in (Zhou et al., 2016), all the layers except for the first convolutional layer and the last fully-connected layer for classification are quantized into 1-bit. We train the baseline binary model for 400 epochs and obtain an accuracy of 85.06%. Based on this pretrained model, we further fine-tune with or without our method.

Table 1. Ablation study on CIFAR-10. For the compared methods, we run them 5 times and show "best (mean±std)".

Method	Acc (%)
Dorefa-Net (Zhou et al., 2016) (Baseline)	85.06
Fine-tuning	85.32 (85.26±0.06)
Ours w/o Noisy-supervision	85.43 (85.36±0.06)
LNS (Ours)	85.78 (85.56±0.11)

Effectiveness of Our Method. To verify the effectiveness of our method, we fine-tune the baseline model without mapping model and our method for 120 epochs with all the same experimental settings as stated in implementation details. In our method, we setting the hyper-parameters as $\alpha = 1.0$ and $\rho = 0.005$. We run them 5 times and show the best, mean and standard values in Table 1. After fine-tuning using our method, we can see that our method without noisy supervision achieves a mean accuracy of 85.36%, adding noisy supervision further improve the accuracy to 85.56%, while simply fine-tuning achieves 85.26%. Both simple fine-tuning and our method can improve the baseline model, but the performance of our method is much better than simple fine-tuning. The results indicate the effectiveness of the proposed binary neuron mapping and the corresponding noisy supervision. The highest accuracy of our method can achieve 85.78%, which is the state-of-the-art as shown in the latter analysis.

We also plot the loss curve and accuracy curve to observe the effect of our method during training. The cross entropy loss curves of simple fine-tuning and our method are shown in Fig. 3(a), and the test accuracy curves of them are shown in Fig. 3(b). The initial loss value and accuracy are 0.37 and 85.06%, respectively, from the pretrained baseline model. At first, we find that Fine-tuning has a much larger loss than our method with the same initial learning rate (lr=0.01), so we decrease the learning rate. Although the loss in Fine-tuning (lr=0.001/0.0001) is decreased, the accuracy on test set has no improvement. From Fig 3(a), we can see that the simple fine-tuning changes the loss at the first as it disturbs the pretrained binary weights largely, while our method does not change the loss much as the mapping model only changes a small portion of the weights. When we decrease the learning rate at the $30th$, $60th$ and $90th$ epoch, the loss values in Fine-tuning and our method will have a relatively large drop. At the last several epochs, simple fine-tuning method with different learning rate achieve a smaller train loss, but a lower test accuracy than our method (Fig. 3(b)). This means that our method can alleviate over-fitting by imposing a noisy supervision on each layer as the noisy weights are often those that over-fit the training data.

From the loss curve and accuracy curve, we know that the training process of our method is more stable than that of simple fine-tuning method. We show the flip rate of the binary weights after each epoch in Fig. 3(c), where flip rate means the ratio of binary weights that are flipped into the opposite values. The flip rate decrease gradually during training in all these curves. The flip rate in Fine-tuning is always higher than our method, which means our method only change a small portion of binary weights which are likely to be noise to explore better performance.

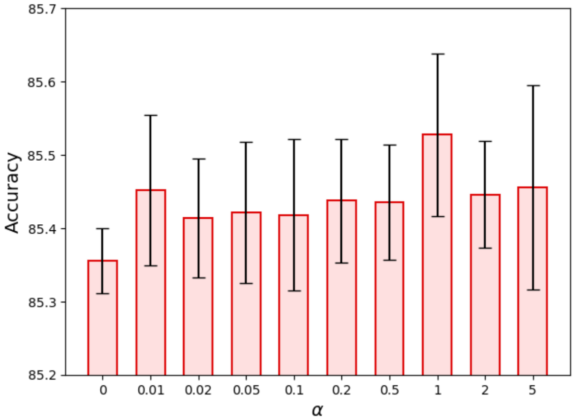


Figure 4. Accuracy v.s. α on CIFAR-10 val set.

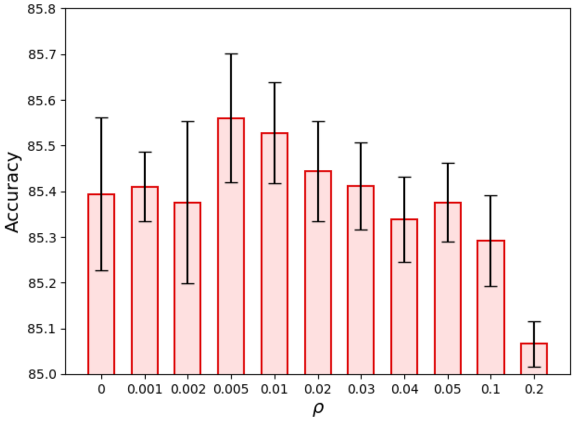


Figure 5. Accuracy v.s. ρ on CIFAR-10 val set.

Analysis of Hyper-parameters. There are two hyper-parameters in our method, *i.e.* α for balancing the cross entropy classification loss and the noisy neuron correction loss, and ρ for controlling the noise rate in the binary weight transformation. We run all the models 5 times and report the mean and std values of the accuracy on CIFAR-10 validation set.

We first fix $\rho = 0.01$ and tune α in range of $\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$ to see the influence of α. The results are shown in Fig. 4. The mean accuracy is 85.35% when $\alpha = 0$, which is higher than the simple fine-tuning. This verifies the effectiveness of the mapping model without self-supervision. When we increase the value of α, the accuracy is improved over that at $\alpha = 0$. The highest mean accuracy occurs around $\alpha = 1$, *i.e.* 85.53%. We can see that our method works at a large range of α and can choose α around 1 for the best performance on CIFAR-10.

For the noise rate ρ, we fix α as 1 and test ρ in $\{0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2\}$. From the results in Fig. 5, we can see that the mean accuracy at different noise rate is different. When ρ is small (around 0.001), the mean accuracy is about 85.4%, which is higher than baseline and simple fine-tuning method. Our method achieve the best mean accuracy at $\rho = 0.005$, which means the ground-truth noise rate is about 0.5%. When ρ is too large, such as $\rho = 0.2$, the mean accuracy drops significantly, even below the simple fine-tuning method. This is due to that there are not so much noise in the pretrained binary weights, setting ρ too large disturbs the good weights and is harmful to the performance.

Table 2. Comparison with SOTA on CIFAR-10. Dorefa-Net and XNOR-Net are implemented by ourselves. For our method, we run it 5 times and show "best (mean±std)" as in (He et al., 2016).

Method	Acc (%)
Dorefa-Net (Zhou et al., 2016) (Baseline)	85.06
XNOR-Net (Rastegari et al., 2016)	85.23
TBN (Wan et al., 2018)	84.34
DSQ (Gong et al., 2019)	84.11
LNS (Ours)	85.78 (85.56±0.11)

Comparison with SOTA. We compare our method with some other state-of-the-art binary neural networks such as Dorefa-Net (Zhou et al., 2016), XNOR-Net (Rastegari et al., 2016), and DSQ (Gong et al., 2019). The results of comparison are listed in Table 2. Note that only the best accuracy is reported for other methods. From the results, our method outperforms the competitors by a large margin and achieves the state-of-the-art result (85.78% accuracy).

4.3. Experiments on ImageNet

In order to validate our method on large-scale dataset, we conduct more experiments on ImageNet classification dataset. Two common network architectures, *i.e.* AlexNet (Krizhevsky et al., 2012) and ResNet-18 (He et al., 2016), are used for experiments.

Effectiveness of Our Method. We test the effect of our method by deploying the proposed method on ResNet-18. The baseline here is the binary ResNet-18 which is quantized using Dorefa-Net (Zhou et al., 2016). We fine-tune the pretrained binary ResNet-18 with or without our method for 60 epochs. These two models use the same hyper-parameter settings. The simple fine-tuning without using our method can improve the Top-1 accuracy to 52.8%. Our method achieves 53.1%, much higher than the baseline and the simple fine-tuning.

Comparison with SOTA. While the ablation study has evaluated the effectiveness of the proposed method, we also compare our method with the state-of-the-art methods to

Table 3. Comparison with SOTA of ResNet-18 architecture on ImageNet. 'W' and 'A' refer to the weight and activation bitwidth, respectively. † represents the result from our implementation.

Method	W	A	Memory	FLOPs	Top-1	Top-5
ResNet-18 (He et al., 2016)	32	32	374 Mbit	1810 M	69.6%	89.2%
BWN (Rastegari et al., 2016)	1	32	34 Mbit	975 M	60.8%	83.0%
HWGQ (Cai et al., 2017)	1	2	34 Mbit	193 M	59.6%	82.2%
TBN (Wan et al., 2018)	1	2	34 Mbit	193 M	55.6%	79.0%
BinaryNet (Hubara et al., 2016)	1	1	28 Mbit	149 M	42.2%	67.1%
Dorefa-Net (Zhou et al., 2016)†	1	1	34 Mbit	163 M	52.5%	76.7%
XNOR-Net (Rastegari et al., 2016)	1	1	34 Mbit	167 M	51.2%	73.2%
Bireal-Net (Liu et al., 2018)	1	1	34 Mbit	163 M	56.4%	79.5%
Bireal-Net (Liu et al., 2018)+PReLU (Baseline)	1	1	34 Mbit	163 M	59.0%	81.3%
PCNN (J=1) (Gu et al., 2019)	1	1	34 Mbit	167 M	57.3%	80.0%
Quantization networks (Yang et al., 2019)	1	1	34 Mbit	163 M	53.6%	75.3%
Bop (Helwegen et al., 2019)	1	1	34 Mbit	163 M	54.2%	77.2%
GBCN (Liu et al., 2019)	1	1	34 Mbit	167 M	57.8%	80.9%
IR-Net (Qin et al., 2020)	1	1	34 Mbit	163 M	58.1%	80.0%
LNS (Ours)	1	1	34 Mbit	163 M	59.4%	81.7%

Table 4. Effectiveness of our method on ResNet-18 architecture on ImageNet.

Method	Top-1	Top-5
Dorefa-Net (Zhou et al., 2016) (Baseline)	52.5%	76.7%
Fine-tuning	52.8%	76.8%
LNS (Ours)	53.1%	77.0%

show the superiority of our method. The compared binary neural network methods include BinaryNet (Hubara et al., 2016), Dorefa-Net (Zhou et al., 2016), XNOR-Net (Rastegari et al., 2016), Bireal-Net (Liu et al., 2018), PCNN (Gu et al., 2019), Bop (Helwegen et al., 2019), GBCN (Liu et al., 2019), *etc*. Two representative 2-bit neural networks, *i.e.* HWGQ (Cai et al., 2017) and TBN (Wan et al., 2018), are also included. Following the common settings (Hubara et al., 2016; Liu et al., 2018), we do not quantize the first convolutional layer and the last fully connected layer for classification.

In ResNet-18 experiments, except for BinaryNet and ABC-Net, all the other methods including our method do not quantize the down-sample layers for fair comparison. The statistics of the compared methods are listed in Table 3. The FLOPs are calculated as real-valued floating-point multiplication plus 1/64 of the amount of 1-bit multiplication as the binary operations including AND and POPCOUNT can be performed in a parallel of 64 by the mainstream CPUs (Liu et al., 2018). We use the ResNet-18 architecture in Bireal-Net as baseline and insert PReLU activation (He et al., 2015) after every binary convolutional layer. This strong baseline has a Top-1 accuracy of 59.0%. Our method is fine-tuned based on the pretrained baseline and finally achieve 59.4% Top-1 and 81.7% Top-5 accuracies, which are higher than the compared models and achieve the state-of-the-art results for binary ResNet-18. It is encouraging to see that our method can beat some methods with 2-bit activations, such as HWGQ (Cai et al., 2017) and TBN (Wan et al., 2018). This gives us the confidence to achieve higher performance with lower bit-width in neural networks.

We also compare our method with several state-of-the-art models for AlexNet architecture which does not has residual connections. from the results in Table 5, tt can be seen that our method outperforms the compared models such as BinaryNet (Hubara et al., 2016), and Dorefa-Net (Zhou et al., 2016), which validates the superiority of our method for different architectures. Moreover, our method with only layer-wise scale factor can achieve higher Top-1 accuracy than XNOR-Net (Rastegari et al., 2016) which uses channel-wise scale factor and more parameters.

Table 5. Comparison with SOTA of AlexNet architecture on ImageNet. †Results from our implementation.

Method	Memory	Top-1
FP32-AlexNet (Krizhevsky et al., 2012)	1860 Mbit	56.6%
BinaryNet (Hubara et al., 2016)	180 Mbit	41.8%
Dorefa-Net (Zhou et al., 2016)	180 Mbit	43.6%
Dorefa-Net (Baseline)†	180 Mbit	43.9%
XNOR-Net (Rastegari et al., 2016)	181 Mbit	44.2%
LNS (Ours)	180 Mbit	44.4%

5. Conclusion

In this paper, we have presented a novel binary neuron mapping method with noisy supervision that leads to state-of-the-art performance for binary neural networks. We apply a learnable mapping model instead of the sign function for weight quantization. The unbiased estimator of mean square error loss is applied to learn from the pretrained binary models. We show that the specially designed loss can converge to ℓ-risk under the clean distribution of binary weights. The experiments on various datasets and neural architectures have verified the effectiveness of the proposed method. The resulted binary neural networks achieve the state-of-the-art performance compared with other approaches.

Acknowledgement

The authors thank the anonymous reviewers for their helpful comments in revising the paper. This work was supported in part by National Key R&D Program of China (2017YFB1002701), NSFC (61632003,61672502), Macau S&T Development Fund (0018/2019/AKP), UM Research Fund (MYRG2019-00006-FST), and in part by the Australian Research Council under Project DE-180101438.

References

Bengio, Y., Leonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv: Learning*, 2013.

Cai, Z., He, X., Sun, J., and Vasconcelos, N. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, 2017.

Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., and Tian, Q. Data-free learning of student networks. In *ICCV*, 2019.

Chen, H., Wang, Y., Xu, C., Shi, B., Xu, C., Tian, Q., and Xu, C. Addernet: Do we really need multiplications in deep learning? In *CVPR*, 2020.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2016.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, pp. 3123–3131, 2015.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In *NeurIPS*, pp. 1269–1277, 2014.

Goldberger, J. and Ben-Reuven, E. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., and Yan, J. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *ICCV*, 2019.

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

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *ICML*, pp. 1737–1746, 2015.

Han, B., Yao, J., Niu, G., Zhou, M., Tsang, I., Zhang, Y., and Sugiyama, M. Masking: A new perspective of noisy supervision. In *NeurIPS*, 2018a.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018b.

Han, K., Guo, J., Zhang, C., and Zhu, M. Attribute-aware attention model for fine-grained representation learning. In *ACM MM*, 2018c.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. Ghostnet: More features from cheap operations. In *CVPR*, 2020.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Helwegen, K., Widdicombe, J., Geiger, L., Liu, Z., Cheng, K.-T., and Nusselder, R. Latent weights do not exist: Rethinking binarized neural network optimization. In *NeurIPS*, 2019.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *NeurIPS*, pp. 4107–4115, 2016.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Lee, K.-H., He, X., Zhang, L., and Yang, L. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2018.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *ICLR*, 2017.

Lin, X., Zhao, C., and Pan, W. Towards accurate binary convolutional neural network. In *NeurIPS*, 2017.

Liu, C., Ding, W., Hu, Y., Zhang, B., Liu, J., and Guo, G. Gbcns: Genetic binary convolutional networks for enhancing the performance of 1-bit dcnns. In *AAAI*, 2019.

Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., and Cheng, K.-T. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 2018.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In *NeurIPS*, 2013.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. In *ICLR*, 2017.

Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., and Song, J. Forward and backward information retention for accurate binary neural networks. In *CVPR*, 2020.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pp. 525–542. Springer, 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

Shen, M., Han, K., Xu, C., and Wang, Y. Searching for accurate binary neural architectures. In *ICCV Workshops*, 2019.

Shen, M., Liu, X., Gong, R., and Han, K. Balanced binary neural networks with gated residual. In *ICASSP*, 2020.

Shu, H., Wang, Y., Jia, X., Han, K., Chen, H., Xu, C., Tian, Q., and Xu, C. Co-evolutionary compression for unpaired image translation. In *ICCV*, 2019.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018.

Vahdat, A. Toward robustness against label noise in training deep discriminative neural networks. In *NeurIPS*, 2017.

Wan, D., Shen, F., Liu, L., Zhu, F., Qin, J., Shao, L., and Tao Shen, H. Tbn: Convolutional neural network with ternary inputs and binary weights. In *ECCV*, 2018.

Wang, Y., Liu, W., Ma, X., Bailey, J., Zha, H., Song, L., and Xia, S.-T. Iterative learning with open-set noisy labels. In *CVPR*, 2018.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, 2019.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., and Hua, X.-s. Quantization networks. In *CVPR*, 2019.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. How does disagreement help generalization against label corruption? In *ICML*, 2019.

Zhang, Z. and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

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