Uncertainty-aware Binary Neural Networks

Junhe Zhao[1] , **Linlin Yang**[2] , **Baochang Zhang**[1*] , **Guodong Guo**[3] and **David Doermann**[4]
[1] Beihang University, Beijing, China
[2]University of Bonn, Germany
[3]Institute of Deep Learning, Baidu Research; National Engineering Laboratory for Deep Learning Technology and Application
[4]University at Buffalo, USA
{jhzhao, bczhang}@buaa.edu.cn

Abstract

Binary Neural Networks (BNN) are promising machine learning solutions for deployment on resource-limited devices. Recent approaches to training BNNs have produced impressive results, but minimizing the drop in accuracy from full precision networks is still challenging. One reason is that conventional BNNs ignore the uncertainty caused by weights that are near zero, resulting in the instability or frequent flip while learning. In this work, we investigate the intrinsic uncertainty of vanishing near-zero weights, making the training vulnerable to instability. We introduce an uncertainty-aware BNN (UaBNN) by leveraging a new mapping function called certainty-sign (c-sign) to reduce these weights' uncertainties. Our c-sign function is the first to train BNNs with a decreasing uncertainty for binarization. The approach leads to a controlled learning process for BNNs. We also introduce a simple but effective method to measure the uncertainty-based on a Gaussian function. Extensive experiments demonstrate that our method improves multiple BNN methods by maintaining stability of training, and achieves a higher performance over prior arts.

1 Introduction

Binary neural networks (BNNs) [Courbariaux *et al.*, 2016] quantize weights and features to single bits and have attracted intense interest for their promising computation acceleration and model compression. Existing BNNs have been applied in many tasks like classification [Rastegari *et al.*, 2016] and detection [Wang *et al.*, 2020]. However, BNNs are still challenged by a drastic drop in performance compared to the full-precision counterparts. One of the main reasons for the performance drop is the discrete weight optimization. The optimization is performed using a non-smooth sign function whose derivative is 0 everywhere except at 0. We denote the point around the zeros as "sensitive points".

To handle these sensitive points, existing BNNs prefer to approximate either the derivative of the sign function or the sign function itself. The straight through estimator (STE) [Bengio *et al.*, 2013] proposes to approximate the derivative of the sign function with the identity for BNNs. Inspired by STE, more precise approximations, including a 1-order approximation [Rastegari *et al.*, 2016] and a polynomial approximation [Liu *et al.*, 2018], were introduced to replace the derivative. These methods provide gradients of sensitive points. In addition to direct derivative approximation, some sign-like functions [Qin *et al.*, 2020; Lin *et al.*, 2020; Liu *et al.*, 2020] have been designed with adaptive parameters or learnable parameters during training to take an asymptotic approach for estimating the sign function. As the training proceeds, those sign-like functions enlarge the gradient of sensitive points to make the gradient large enough to change the binary weights.

However, the existing derivative or sign function approximation methods all emphasize the gradient magnitude of the sensitive points but ignore the optimization direction. The sign may provide the unstable optimization direction because of the instability of sensitive points. It is obvious that weights near zero are more uncertain and are thus vulnerable and un-

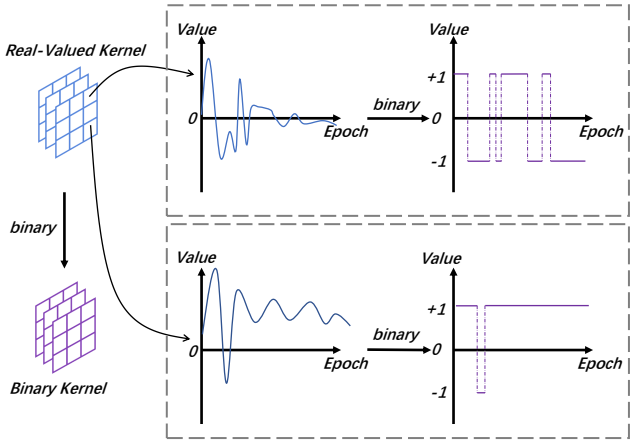


Figure 1: An illustration of the weight fluctuation during the training process of BNNs is shown here. The diagram above corresponds to an unstable real-valued weight, leading to an uncertain binarized state. In contrast, the weight in the diagram below is relatively mild and results in a stable state.

*Baochang Zhang is the corresponding author.

stable during binarization. As shown in Fig. 1, the weights that fluctuate around zero provide unstable optimization directions. Learning with uncertain direction will induce slow convergence and instability for BNNs.

Helwegen *et al.* [Helwegen *et al.*, 2019] proposes to directly optimize the binary weights according to the gradient and skip the update of full-precision auxiliary weights. However, this approach is ineffective when attempting to estimate the gradient required for sign flipping. Instead, we propose to model the uncertainty of the binarization and determine the optimization direction based on the uncertainty. The uncertainty can be used to determine which weights are hard to binarize. It reflects the reliability of the optimization.

In this paper, we analyze the influence of uncertainty introduced by weight binarization and present an uncertainty-aware BNN (UaBNN) to learn BNNs by minimizing the uncertainty of binarization. More specifically, we describe a simple yet effective method based on a Gaussian function to quantitatively measure uncertainty in BNNs. A new certainty-sign (c-sign) function is introduced to reduce the uncertainty, improve stability during training. Our contributions can be summarized as follows:

- We introduce a Gaussian function to quantitatively measure the uncertainty in BNNs. We further show that uncertainty is caused by weight binarization.
- We design a new certainty-sign function (c-sign) to control the binarization during training. We further propose an uncertainty-aware BNN (UaBNN) to learn BNNs with a decreasing uncertainty.
- Extensive experiments illustrate that our method improves multiple BNN methods by maintaining stability during training.

2 Related Work

Very low bit-width quantization on the model will inevitably cause information loss due to the quantization error. Many approaches have been proposed to alleviate the information loss, such as minimizing the distance between the real-valued weight and the binarized weight [Rastegari *et al.*, 2016] or adjusting the distribution of parameters to reduce the quantization error [Gu *et al.*, 2019b]. However, there still exists a nontrivial accuracy gap between BNN and its full-precision counterpart. This accuracy gap mainly comes from two aspects, the limited representation capacity of low bitwidth and the difficult optimization of a non-smooth sign function.

Introducing extra feature information, and therefore increasing the model capacity, is one effective way to improve the performance of BNNs. XNOR-Net [Rastegari *et al.*, 2016] adds a layer-wise "scale factor" to reconstruct the binarized kernels and achieves a better approximation. Inspired by this, XNOR-Net++ [Bulat and Tzimiropoulos, 2019] further applies three learnable scale factors corresponding to different dimensions of the feature maps of 1-bit convolution. Besides adding more scale factors, Real-To-Binary [Martinez *et al.*, 2019] introduces a gating module like SE-Net [Hu *et al.*, 2018] to rescale the feature maps of the channels before binarization.

Another way to improve the performance of BNNs is by adopting a proper optimization method for the quantization. Inspired by STE [Courbariaux *et al.*, 2016], most existing works update the parameters approximately and introduce auxiliary loss functions. Gu *et al.* [Gu *et al.*, 2019a] reformulates the optimization using a projection function from a discrete backpropagation view and proposes projection convolution neural networks. BONN [Gu *et al.*, 2019b] minimizes the quantization error based on the Bayesian method and redistributes the real weights to a bimodal distribution. IR-Net [Qin *et al.*, 2020] introduces the information entropy loss and optimized it with quantization error simultaneously. Our work also falls into this line. We investigate the intrinsic uncertainty nature behind the weight binarization and propose c-sign and UaBNN to learn binarized neural networks with improved stability, and higher accuracy.

3 Method

In this section, we first introduce BNNs briefly and then present the details of uncertainty in weight binarization. Finally, we propose our UaBNN that minimizes the uncertainty of BNNs during training.

3.1 Preliminary

In the L-layer CNN, we denote the weights and features of the l-th layer as $\boldsymbol{W}^l$ and $\boldsymbol{F}^l$. The operation in the l-th layer is expressed as:

$$\boldsymbol{F}^{l+1} = \phi^l(\boldsymbol{W}^l * \boldsymbol{F}^l), \tag{1}$$

where $*$ denotes convolution operation, and $\phi^l(\cdot)$ represents other operations in the l-th layer, such as BatchNorm, ReLU, and else for simplicity. In BNNs, each element of $\boldsymbol{W}^l$ and $\boldsymbol{F}^l$ are mapped to $\{+1, -1\}$ by sign function, *i.e.* , $\boldsymbol{W}_B^l = sign(\boldsymbol{W}^l)$, $\boldsymbol{F}_B^l = sign(\boldsymbol{F}^l)$.

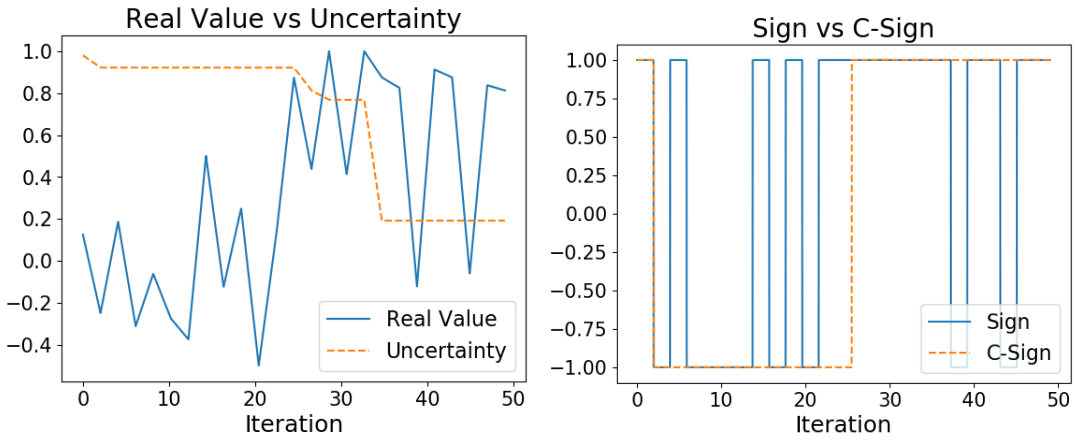
However, the sign function discards the amplitude information of variables and leads to large quantization errors. To alleviate the precision loss, the scale factor [Rastegari *et al.*, 2016] is used to enhance the representation ability of the 1-bit network and widely used in existing BNNs methods while keeping the weights and features still binarized. And the operations in l-th layer can be represented as:

$$\boldsymbol{F}^{l+1} = \phi^l(\boldsymbol{\alpha}^l \cdot (\boldsymbol{W}_B^l \circledast \boldsymbol{F}_B^l)), \tag{2}$$

where $\circledast$ stands for a binary convolution consisting of XNOR, and popcount operations, and scale factor is denoted as $\boldsymbol{\alpha}^l$. In this way, the operands in convolution are turned into $\{-1, +1\}$, and thus the multiply-accumulation operations in real-weight convolution can be replaced with light-weighted XNOR and popcount operations for a simplified convolution [Courbariaux *et al.*, 2016] to accelerate computation and reduce the storage.

3.2 Uncertainty in BNN

Conventional methods utilize the sign function for mapping the continuous weight value to a discrete state, whereas different full-precision networks correspond to an identity binary network and lead to the same performance in forward propagation, returning the same gradients. This is an obstacle to the optimization of BNNs. Moreover, the continuous



(a) (b)

Figure 2: (a) An example of real value and its corresponding uncertainty based on Eq. 5, (b) The binarization results of the example using the sign and the c-sign functions with $m = 2$.

weight values around zero are clearly unstable. An alternative is to calculate the discrete state based on the uncertainty instead of the continuous value - this is much reliable and stable. In this case, we first introduce the concept of uncertainty for BNNs, which can distinguish BNNs even if they perform the same in forward propagation, and then introduce our uncertainty aware optimization for BNNs.

Intuitively, the sign of weights close to zero may frequently flip in the training process, leading to the state more uncertain. In contrast, the state that is far from zero weight is more stable and thus more certain. In order to quantitatively estimate the uncertainty of BNNs, we introduce a novel function to estimate the uncertainty of weight binarization by considering the following characteristics: the uncertainty is maximum at 0 and decreases gradually as the weights approach +1/-1. Using the predicted continuous value x ($-1 \leq x \leq 1$) and its target (+1 and -1), we model the uncertainty as below:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2}), \tag{3}$$

with σ as a hyperparameter. For simplicity, we also denote it as an element-wise function for matrices. We apply this Gaussian function to formulate the uncertainty of BNNs. A binarization with a higher score means a lower confidence and a larger potential for reversal. Although Eq. 3 provides a measure of uncertainty for static BNNs. For better optimization, it is also necessary to consider the fluctuation of uncertainty in the dynamic BNNs training. We not only aim to obtain a certainty state this time but also keep a stable training process and avoid outliers. To this end, we comprehensively estimate the uncertainty by bringing in the last m state of BNNs as:

$$\hat{f}(x_t) = \begin{cases} f(x_t) & t \leq m \\ 1 - \prod_{i=t-m}^{t} (1 - f(x_i)) & t > m, \end{cases} \tag{4}$$

where t denotes t-th iteration. In this way, we dynamically calculate the uncertainty of the BNNs in the training process.

3.3 Uncertainty aware Binary Neural Networks

To minimize the uncertainty of BNNs, we propose a certainty-sign (c-sign) function. Considering the t-th iteration in training, c-sign can be represented as:

$$csign(x_t) = \begin{cases} sign(x_t) & \hat{f}(x_t) \leq \max(\hat{f}(x_{t-1}), \Delta) \\ csign(x_{t-1}) & otherwise, \end{cases} \tag{5}$$

where x_t is the full-precision weight and $\hat{f}(x_t)$ is calculated as Eq. 4. Δ here is a threshold for uncertainty. For convenience, instead of a certain value, we introduce Δ as an adaptive threshold for uncertainty. In details, the value of Δ in the following section corresponds to the percent of the sorted values in each layer. And excessive Δ may cause the c-sign function to degenerate to sign function. By replacing the sign function with c-sign function, the uncertainty of binarization can be theoretically proven to decrease, as shown in Theorem 1. An example of sign and c-sign is shown in Fig. 2.

Theorem 1. *If $\hat{f}(x_t) \leq \hat{f}(x_{t-1})$ is satisfied throughout the training, $f(x_t)$ is decreasing with m intervals.*

Proof.

$$\begin{aligned} &\hat{f}(x_t) - \hat{f}(x_{t-1}) \\ &= \prod_{i=t-m-1}^{t-1} (1 - f(x_i)) - \prod_{i=t-m}^{t} (1 - f(x_i)) \\ &= (f(x_t) - f(x_{t-m-1})) \prod_{i=t-m-1}^{t-1} (1 - f(x_i)) \end{aligned} \tag{6}$$

Since

$$0 \leq f(\cdot) \leq 1,$$

and x is updated only if

$$\hat{f}(x_t) \leq \hat{f}(x_{t-1}),$$

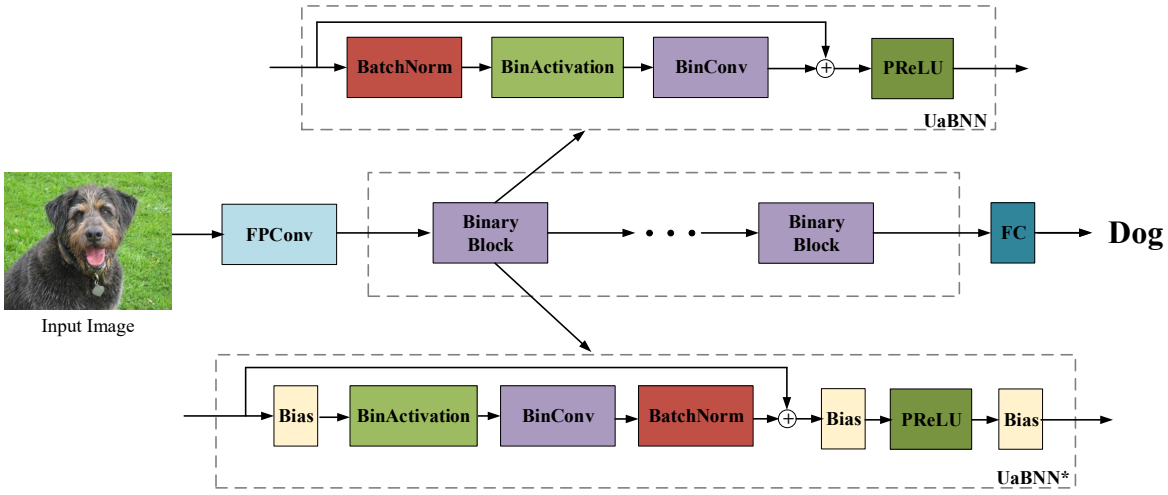


Figure 3: Network architectures of UaBNN and UaBNN*. For UaBNN, in both WRN22 and ResNet18, we replace the original block with illustrated block, following the same structure as [Gu *et al.*, 2019b] but modify the kernel quantization method with the c-sign function. For UaBNN*, the sign function is replaced by c-sign function when binarizing the weights, while other structures setting are consistent with ReActNet [Liu *et al.*, 2020].

based on Eq. 5, we can get that $f(x)$ is decreasing with m intervals. That is to say, along with t increase, $f(x)$ will decrease at each m intervals. □

In this way, we can denote the uncertainty of the binary network at the t-th iteration as $\sum \hat{f}(\boldsymbol{W}_t)$, *i.e.* , the sum of all the uncertainty of weights at t-th iteration. Then in the next iteration, the element of $\boldsymbol{W}_t$ will be updated according to Eq.5, its state at $t+1$ iteration will be retained as the former state. More specifically, we apply an asynchronous update during training. In the forward propagation, we binarize the full-precision weights following Eq. 5. Some of the binarization weights will not be updated because of their uncertainty. In the backpropagation, the gradients are utilized to update the corresponding full-precision weights. Note that the gradients of full-precision weights may be unmatched to their full-precision weights' actual values. In this way, we ensure the uncertainty of BNN decrease. A detailed process is presented in Alg. 1.

Algorithm 1 Uncertain aware Binary Neural Network

Input:
The full-precision weights W; the input dataset.
Output:
UaBNN with the updated W.
Initialize W randomly;
repeat
  // Forward propagation
  for $l = 1$ to L **do**
    Calculate the uncertainty based on Eq. 3;
    Modify Δ and binarize weights W_B^l use Eq. 5;
    Perform activation binarization F_B^l; // Using the sign function
    Perform 2D convolution with W_B^l
  end for
  // Backward propagation
  for $l = L$ to 1 **do**
    Compute gradients based on the binarization weights W_B^l
    Update full-precision weights W^l
  end for
until convergence

4 Experiments

In this part, we investigate the effectiveness of the proposed method on CIFAR-10/100 [Krizhevsky and others, 2009] and ILSVRC12 ImageNet [Deng *et al.*, 2009] datasets with the mainstream deep CNN architectures, including ResNet [He *et al.*, 2016] and Wide ResNet [Zagoruyko and Komodakis, 2016]. Firstly, we elaborate the experiment setups in Section 4.1, including datasets, models, as well as hyperparameter settings. In section 4.2, a comprehensive comparison on both the CIFAR and the ImageNet datasets in terms of accuracy is illustrated. Finally, we further analyze the effects of the proposed c-sign method during the training process in Section 4.3.

4.1 Datasets and Implementation Details

CIFAR is a small-scale natural image classification dataset, with the color image size of 32×32. The training set and testing set of CIFAR10/100 are composed of 50,000 pictures and 10,000 pictures, respectively, across the 10/100 classes. Moreover, ILSVRC12 ImageNet is a more challenging and diverse dataset, which contains 1.2 million training images and 50,000 validation images across 1000 classes. Its large scale and high resolution make it a harder task compared to CIFAR.

For the CIFAR10/100 dataset, we employ Wide ResNet (WRN) to verify the superiority and effectiveness of our

Model	Kernel-Stage	Method	#Param	W/A	CIFAR10(%)	CIFAR100(%)
WRN22	16-16-32-64	XNOR-Net[Rastegari *et al.*, 2016]	0.27M	1/1	81.90	53.17
		Bi-Real Net[Liu *et al.*, 2018]	0.27M	1/1	85.16	57.34
		BONN[Gu *et al.*, 2019b]	0.27M	1/1	87.34	60.91
		UaBNN	0.27M	1/1	88.03	61.68
		FP32	0.27M	32/32	91.66	67.51
WRN22	64-64-128-256	Bi-Real Net[Liu *et al.*, 2018]	4.3M	1/1	90.65	68.51
		PCNN[Gu *et al.*, 2019a]	4.3M	1/1	91.37	69.98
		BONN[Gu *et al.*, 2019b]	4.3M	1/1	92.36	-
		UaBNN	4.3M	1/1	93.37	72.01
		FP32	4.3M	32/32	95.75	77.34

Table 1: Test accuracies on CIFAR Dataset. 'W' and 'A' refer to the weight and activation bitwidth, respectively. 'FP' denotes the full-precision model. The backbone of all of the models is WRN22.

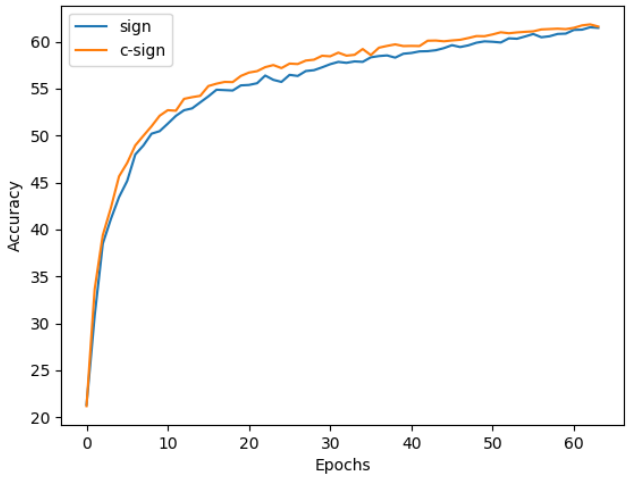


Figure 4: The accuracy of ReActNet using sign and csign on ImageNet. The backbone of the two networks is ResNet18. We can see that ReActNet with csign achieves higher accuracy and faster convergence.

UaBNNs. In contrast, we adopt ResNet18 to evaluate our UaBNNs for the ImageNet dataset. For ResNet18, we binarizes the features and kernels in the backbone convolution layer except for the first and last layers full-precision, the same with other BNNs methods. Moreover, we also apply the modified shortcut method in Bi-Real Net, which uses additional shortcut layers in each block and keep them real-valued. The learning rate is initially set to 0.001, with an Adam optimizer of momentum 0.9. A linear decay strategy is employed for the learning rate, which degrades the learning rate in a linear manner.

WRN is a network taking the ResNet as the prototype but introduce a new depth factor k to adjust the feature map depth expansion through 3 stages, where the spatial dimension of the features remains the same. For brevity, we set k to 1 in the following experiments. The number of channels in the first stage is another important parameter in WRN. We set it to 16 and 64, and thus result in network configurations with 16-16-32-64 and 64-64-128-256, respectively. The learning rate is initially set to 0.1, with an SGD optimizer of momentum 0.9, and we also apply a cosine annealing decay methods. And other training settings are the same as those described in [Gu *et al.*, 2019b]. We denote WRN-22 as a WRN network with 22 convolutional layers. The network architectures of our UaBNN and UaBNN* are shown in Fig. 3.

As for the hyperparameters introduced in UaBNN, m is set to 2. σ is influenced by the initialize method of corresponding parameters, and we set it equals to the variance of initialization parameters. For Δ, which controls the stability of training and adaptively varies in the training, we test it from (0.05, 0.5), and set it to 0.1 for a relative rate for a better performance.

4.2 Results

Firstly, we evaluate our UaBNN on CIFAR with WRN in two different configurations, 16-16-32-64 and 64-64-128-256. Moreover, we report the accuracy of the full-precision counterpart. We train the BNNs in our UaBNN manner, following the settings as depicted in Section 4.1. Data augmentation is applied during training. The images are padded with a size of 4 and are randomly divided into 32×32 windows for CIFAR-10/100. Table 1 illustrates that our UaBNN outperforms other BNNs on both CIFAR-10 and CIFAR-100 datasets, which indicates the advantage of our method in building 1-bit CNNs.

On the ImageNet dataset, we further evaluate the performance of our method on the ImageNet dataset. Notably, we adopt two data augmentation methods in the training set: 1) cropping the image to the size of 224×224 at random locations, and 2) flipping the image horizontally. In the test set, we simply crop the image to 224×224 from the center. We use ResNet18 as the backbone and only slightly adjust the structure following [Gu *et al.*, 2019b]. We compare our UaBNN with other state-of-the-art quantized networks, including TBN [Wan *et al.*, 2018], BNN [Courbariaux *et al.*, 2016], XNOR-Net [Rastegari *et al.*, 2016], ABC-Net [Lin *et al.*, 2017], Bi-Real Net [Liu *et al.*, 2018], PCNN [Gu *et al.*, 2019a], IR-Net [Qin *et al.*, 2020], BONN [Gu *et al.*, 2019b], and RBNN [Lin *et al.*, 2020]. Table 2 indicates that UaBNN achieves a superior performance among these 1-bit CNNs. From Table 2, we can see that our method makes full usage of uncertainty, and the improvements are up to 1.0% Top-1 accuracy and 0.6% Top-5 accuracy. The results show that UaBNN is not limited to small datasets, but also works well

Model	W	A	Top-1	Top-5
ResNet18 [He *et al.*, 2016]	32	32	69.3	89.2
TBN [Wan *et al.*, 2018]	1	2	55.6	79.0
BNN [Courbariaux *et al.*, 2016]	1	1	42.2	67.1
XNOR-Net [Rastegari *et al.*, 2016]	1	1	51.2	73.2
ABC-Net [Lin *et al.*, 2017]	1	1	42.7	67.6
Bi-Real Net [Liu *et al.*, 2018]	1	1	56.4	79.5
PCNN [Gu *et al.*, 2019a]	1	1	57.3	80.0
IR-Net [Qin *et al.*, 2020]	1	1	58.1	80.0
BONN [Gu *et al.*, 2019b]	1	1	59.3	81.6
RBNN [Lin *et al.*, 2020]	1	1	59.6	81.6
UaBNN	1	1	60.6	82.2
ReActNet* [Liu *et al.*, 2020]	1	1	61.4	83.2
UaBNN*	1	1	61.9	83.4

Table 2: Test accuracies on ImageNet. 'W' and 'A' refer to the weight and activation bitwidth, respectively. The backbone of all the models is ResNet18. ReactNet* means a quick version of original implement for 64 epoch from scratch.

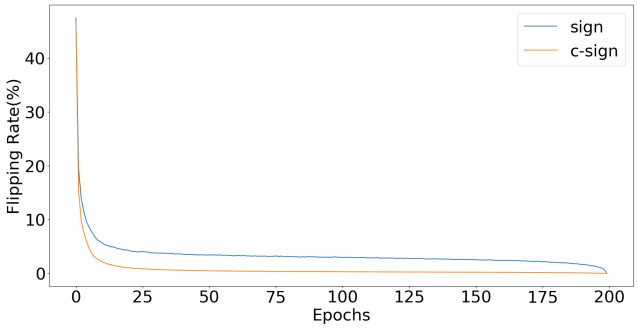


Figure 5: The flip rate of binary weights, *i.e.* the frequency of sign changes. Big value of flipping rate means more frequent flips. UaBNN can adjust the flip rate and therefore control the stability of training.

on the large dataset. This further verifies the generalization capability of the proposed UaBNN. We also compare with ReActNet [Liu *et al.*, 2020] on the ImageNet dataset. ReActNet achieves the best performance on BNNs so far, with the accuracy of 65.9% on ImageNet. It introduces some full-precision parameters and a two-steps training strategy with 512 epochs in total, which is computation-consuming. Here, we evaluate our UaBNNs on a relatively quick version of ReActNet, which is trained from scratch for 64 epochs for a fair comparison with other methods. Our method still achieves an impressive 0.5% improvement, which further verifies the effectiveness of our method.

4.3 Analysis

In this part, we analyze the influence of our proposed c-sign function on the uncertainty and stability of BNNs. Specifically, to understand the c-sign better, we introduce the flipping rate, which denotes the proportion of reversed binary parameters. Flipping rate reflects the uncertainty of binary neural networks in a more intuitive way that a lower flipping rate corresponds to a more certain BNNs. In Fig. 5, we illustrate the comparison of sign and c-sign on flipping rate on CIFAR-100 dataset. During the training process, our c-sign can effectively reduce the uncertainty of BNNs and, consequently, the training of BNNs are more stable, leading to a decreasing flipping rate and a better BNN.

5 Conclusions

In this paper, we have proposed and described Uncertainty-aware Binary Neural Networks (UaBNN), which firstly takes the uncertainty of BNNs into consideration, resulting in a unified uncertain-aware framework. The uncertainty in BNNs is a novelty index used to measure the training stability of BNN. Comprehensive studies on the uncertainty have been conducted. Extensive experiments on CIFAR and ImageNet demonstrate that UaBNN achieves effective enhancement on multiple BNNs methods for WRNs and ResNet18. In our future work, we will use our method to improve BNN models for other applications, such as visual object detection and tracking.

Acknowledgements

This study was supported by Grant NO.2019JZZY011101 from the Key Research and Development Program of Shandong Province to Dianmin Sun. This work was supported in part by the National Natural Science Foundation of China under Grant 62076016 and 61876015.

References

[Bengio *et al.*, 2013] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[Bulat and Tzimiropoulos, 2019] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural

networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

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

[Gu *et al.*, 2019a] Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *AAAI Conference on Artificial Intelligence*, 2019.

[Gu *et al.*, 2019b] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Jianzhuang Liu, Guodong Guo, and Rongrong Ji. Bayesian optimized 1-bit cnns. In *IEEE International Conference on Computer Vision*, pages 4909–4917, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Helwegen *et al.*, 2019] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in Neural Information Processing Systems*, pages 7533–7544, 2019.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[Krizhevsky and others, 2009] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. *Technical report, Citeseer*, 2009.

[Lin *et al.*, 2017] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[Lin *et al.*, 2020] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Advances in Neural Information Processing Systems*, 33, 2020.

[Liu *et al.*, 2018] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *European Conference on Computer Vision*, pages 747–763. Springer, 2018.

[Liu *et al.*, 2020] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision*, pages 143–159. Springer, 2020.

[Martinez *et al.*, 2019] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2019.

[Qin *et al.*, 2020] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2250–2259, 2020.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[Wan *et al.*, 2018] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *European Conference on Computer Vision*, pages 315–332, 2018.

[Wang *et al.*, 2020] Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2049–2058, 2020.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*. British Machine Vision Association, 2016.