Binary Stochastic Flip Optimization for Training Binary Neural Networks

Tatsukichi Shibuya[1], Nakamasa Inoue[1], Rei Kawakami[2], Ikuro Sato[1,3]
[1]*Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan.*
[2]*Department of Systems and Control Engineering, Tokyo Institute of Technology, Tokyo, Japan.*
[3]*Denso IT Laboratory, Tokyo, Japan.*

***Abstract*—For deploying deep neural networks on edge devices with limited resources, binary neural networks (BNNs) have attracted significant attention, due to their computational and memory efficiency. However, once a neural network is binarized, finetuning it on edge devices becomes challenging because most conventional training algorithms for BNNs are designed for use on centralized servers and require storing real-valued parameters during training. To address this limitation, this paper introduces binary stochastic flip optimization (BinSFO), a novel training algorithm for BNNs. BinSFO employs a parameter update rule based on Boolean operations, eliminating the need to store real-valued parameters and thereby reducing memory requirements and computational overhead. In experiments, we demonstrated the effectiveness and memory efficiency of BinSFO in finetuning scenarios on six image classification datasets. BinSFO performed comparably to conventional training algorithms with a 70.7% smaller memory requirement.**

I. Introduction

There is a high demand for implementing deep neural networks on memory-constrained edge devices. Binary neural networks (BNNs) have become a promising solution due to their ability to significantly reduce model size and computational complexity [1]–[7]. The core concept of BNNs is to represent activations and parameters by binary values (*e.g.*, ± 1) so that test-time inference solely relies on computationally efficient Boolean operations.

For training BNNs, a number of studies have proposed techniques for improving effectiveness and efficiency. The straight-through estimator (STE) [8] is the most commonly used technique; it allows the backpropagation of gradients with non-differentiable operations, such as the sign function for binarization, through approximated gradients. Follow-up studies have explored more effective estimators, such as those implemented in EDE [9], DSQ [10], FDA [11], RBNN [12], and ReSTE [13]. In these methods, real-valued parameters are quantized to produce binary ones in the feedforward process and updated by approximate gradients using STE. The need to keep real parameters in training demands considerable memory usage, limiting the potential for finetuning BNNs on memory-constrained edge devices with small datasets.

Some studies have discussed training BNNs with discrete optimization algorithms, such as those for mixed integer programming and constraint programming [14]–[16]. However, these algorithms are non-stochastic and the memory requirements increase with the size of the training dataset, making their application impractical even for small datasets such as MNIST.

To address this limitation, this paper proposes binary stochastic flip optimization (BinSFO), a stochastic training algorithm that does not require storing real-valued parameters during training. Specifically, BinSFO introduces the binary parameter update rule based on Boolean operations and applies it iteratively for training BNNs. In experiments, we demonstrated the effectiveness and memory efficiency of BinSFO in finetuning scenarios on six image classification datasets. BinSFO performed comparable to ReSTE [13] under a 70.7% reduction in the memory requirements.

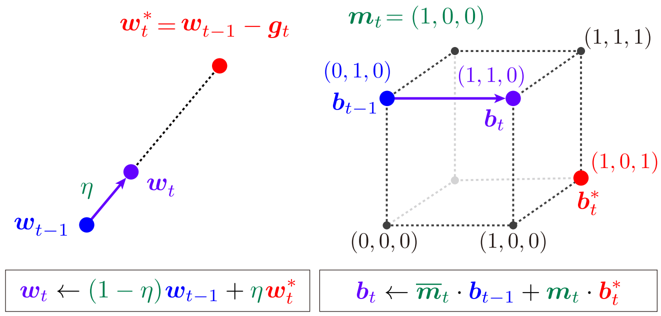
II. Related Work

BNNs [1] have been studied from both architectural and optimization perspectives. Convolutional architectures are the major architecture. Examples include XOR-Net utilizing channel-wise scaling [17], ABC-Net with weight approximation [18], Bi-RealNet incorporating shortcut propagation [19], and ReActNet with the RPReLU activation function [20]. Recently, transformer- and MLP-based architectures such as BinaryViT [21], BiT [22], and BCDNet [23] have also been demonstrated to be effective.

Optimization techniques and gradient estimators have been investigated to bridge the gap between continuous and binary values [8]–[13]. For example, ReSTE [13] is a function-based estimator that flexibly balances the estimation error and gradient stability. Traditional optimization methods, such as SGD and Adam [24], are widely used for training BNNs. There have also been studies discussing their effectiveness and extensions, such as AdamBNN [25] and Bop [26]. In most studies, real-valued parameters are stored during training to perform backpropagation with approximated gradients, and this constraint has traditionally been considered necessary. However, in the context of finetuning with small datasets, this may not always be required, motivating us to explore a training algorithm based on Boolean operations.

III. Proposed Method

This section introduces BinSFO, a training algorithm for BNNs developed in this study having a parameter update rule based on Boolean operations, which eliminates the need to store real-valued parameters during training.



(a) SGD in the real space (b) BinSFO in the Hamming space

Fig. 1: Comparison of parameter update rules.

Algorithm 1: BinSFO

Requires: *L-layer BNN F, Initial parameters $\{\boldsymbol{b}_0^{(l)}\}_{l=1}^{L}$, Training dataset $\mathcal{D}_{\text{train}}$, Hypermask distribution P, Loss $\mathcal{L}$.*

for $t = 1$ **to** T **do**

  $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}_{\text{train}}$ # Draw a mini-batch

  $\hat{\boldsymbol{y}} \leftarrow F(\boldsymbol{x})$ # Forward pass

  for $l = L$ **to** 1 **do**

    $\boldsymbol{g}_t^{(l)} \leftarrow \nabla_{\boldsymbol{b}^{(l)}} \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ # Compute the gradient

    $\boldsymbol{b}_t^{(l),*} \leftarrow [\![\boldsymbol{g}_t^{(l)} \leq 0]\!]$ # Compute the target

    $\boldsymbol{m}_t^{(l)} \sim P(\boldsymbol{m}_t | \boldsymbol{b}_{t-1}, \boldsymbol{g}_t)$ # Draw a hypermask

    $\boldsymbol{b}_t^{(l)} \leftarrow \overline{\boldsymbol{m}}_t^{(l)} \cdot \boldsymbol{b}_{t-1}^{(l)} + \boldsymbol{m}_t^{(l)} \cdot \boldsymbol{b}_t^{*,l}$

return $\{\boldsymbol{b}_T^{(l)}\}_{l=1}^{L}$

A. Notation and settings

Let $\boldsymbol{b} \in \mathbb{H}^N$ be a binary vector that represents flattened learnable parameters of a BNN, where $\mathbb{H}^N = \{0,1\}^N$ is a Hamming space and N is the number of parameters. While most previous studies use $\mathbb{B} = \{1,-1\}$ to denote binary parameters, we use $\mathbb{H} = \{0,1\}$ to make the discussion concise with Boolean operations: OR $a+b$, AND $a \cdot b$, and NOT $\overline{a}$ for $a, b \in \mathbb{H}$. We consider an iterative algorithm with the iteration index $t \in \mathbb{N}$ and denote by $\boldsymbol{b}_t \in \mathbb{H}^N$ the parameters at t.

B. Algorithm

Updating binary parameters. The parameter update rule of BinSFO is derived by constraining the update rule of SGD, defined in the real space $\mathbb{R}^N$, to the Hamming space $\mathbb{H}^N$. With SGD, real-valued parameters $\boldsymbol{w}_t \in \mathbb{R}^N$ are updated at each iteration by the following rule:

$$\boldsymbol{w}_t \leftarrow (1-\eta)\boldsymbol{w}_{t-1} + \eta \boldsymbol{w}_t^*, \tag{1}$$

where $\boldsymbol{w}_t^* = \boldsymbol{w}_{t-1} - \boldsymbol{g}_t$ is a target, $\boldsymbol{g}_t$ is a loss gradient, and η is a hyperparameter. When $\eta \in [0,1]$, this update rule is equivalent to solving the following minimization problem:

$$\boldsymbol{w}_t = \underset{\boldsymbol{p} \in \mathbb{R}^N}{\operatorname{argmin}} \left(\|\boldsymbol{w}_{t-1} - \boldsymbol{p}\|_2 + \|\boldsymbol{w}_t^* - \boldsymbol{p}\|_2 \right). \tag{2}$$

This is because $\boldsymbol{w}_t$ is an interpolation of $\boldsymbol{w}_{t-1}$ and $\boldsymbol{w}_t^*$ as shown in Figure 1a. Note that the solution of this minimization problem exists as the set of points $\mathcal{W}_t \subset \mathbb{R}^N$ given by

$$\mathcal{W}_t = \{\boldsymbol{w}_t : \boldsymbol{w}_t = (1-\eta)\boldsymbol{w}_{t-1} + \eta \cdot \boldsymbol{w}_t^*, \quad \eta \in [0,1]\}, \tag{3}$$

and specifying a value for η yields a unique solution in $\mathcal{W}_t$.

BinSFO replaces $\mathbb{R}^N$ in Eq. (2) with $\mathbb{H}^N$, giving rise to the following binary optimization problem:

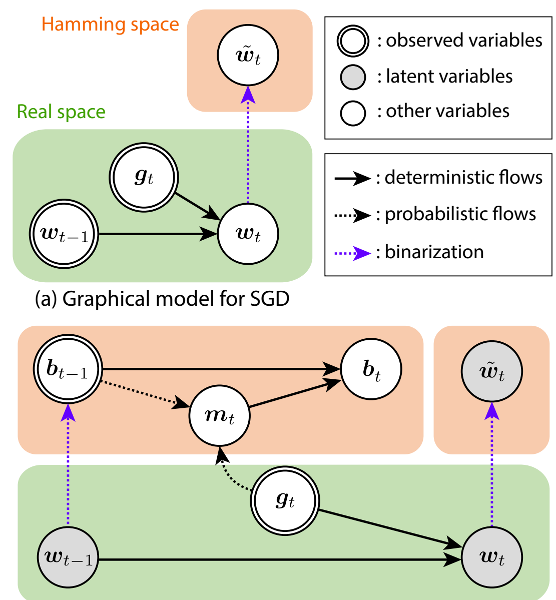
$$\boldsymbol{b}_t = \underset{\boldsymbol{p} \in \mathbb{H}^N}{\operatorname{argmin}} \left(d(\boldsymbol{b}_{t-1}, \boldsymbol{p}) + d(\boldsymbol{b}_t^*, \boldsymbol{p}) \right), \tag{4}$$

where $d(\boldsymbol{a}, \boldsymbol{b}) = |\{i : a_i \neq b_i\}|$ is the Hamming distance. The solution exists as the set of points $\mathcal{B}_t \subset \mathbb{H}^N$ given by

$$\mathcal{B}_t = \{\boldsymbol{b}_t : \boldsymbol{b}_t = \overline{\boldsymbol{m}} \cdot \boldsymbol{b}_{t-1} + \boldsymbol{m} \cdot \boldsymbol{b}_t^*, \quad \boldsymbol{m} \in \mathbb{H}^N\}. \tag{5}$$

In analogy to SGD, specifying values for $\boldsymbol{m}$ yields a unique solution as shown in Figure 1b. Consequently, BinSFO uses the following rule to update binary parameters:

$$\boldsymbol{b}_t \leftarrow \overline{\boldsymbol{m}}_t \cdot \boldsymbol{b}_{t-1} + \boldsymbol{m}_t \cdot \boldsymbol{b}_t^*. \tag{6}$$



(b) Graphical model for BinSFO

Fig. 2: Graphical models.

This rule ensures that $\boldsymbol{b}_t \in \mathcal{B}_t$, but it requires the determination of $\boldsymbol{b}_t^* \in \mathbb{H}^N$ and $\boldsymbol{m}_t \in \mathbb{H}^N$, which we refer to as the target and hypermask, respectively.

Target. We define the target as $\boldsymbol{b}_t^* = [\![\boldsymbol{g}_t \leq 0]\!]$, where $[\![P]\!]$ takes 1 if the proposition P is true and 0 otherwise in an element-wise manner. This helps reduce the loss corresponding to the mini-batch at each iteration because updating parameters in the direction opposite to the gradient can decrease loss. This step temporarily requires calculating the gradient $\boldsymbol{g}_t$ in the real space, but not storing real-valued parameters.

Hypermask. We sample a hypermask $\boldsymbol{m}_t$ from a probabilistic distribution $P(\boldsymbol{m}_t | \boldsymbol{b}_{t-1}, \boldsymbol{g}_t)$ at each iteration as detailed in the next subsection.

Algorithm. The learning procedure of BinSFO is summarized in Algorithm 1. It applies the parameter update rule in Eq. (6) for each layer of a BNN.

C. Hypermask sampling

We define the hypermask distribution to bridge the gap between SGD and BinSFO. Specifically, we derive the dis-

TABLE I: Full fine-tuning results. Test accuracies (%) are reported for each dataset. Bit-widths for weights are the values for training/testing. Total memory required for training relative to ReSTE [13] is shown in the last column.

	Method	Bit-widths	CIFAR-10	CIFAR-100	Caltech-101	Caltech-256	Flowers	Pets	Average	Total memory (%)
ReActNet	STE	32/1	$92.29_{\pm 0.23}$	$75.19_{\pm 0.03}$	$94.25_{\pm 0.30}$	$81.71_{\pm 0.33}$	$96.46_{\pm 0.16}$	$89.24_{\pm 0.52}$	88.19	68.1
	Bop	32/1	$92.36_{\pm 0.28}$	$74.67_{\pm 0.06}$	$94.31_{\pm 0.35}$	$81.75_{\pm 0.27}$	$96.27_{\pm 0.11}$	$89.59_{\pm 0.74}$	88.16	71.9
	ReSTE	32/1	$\mathbf{92.64}_{\pm 0.10}$	$75.33_{\pm 0.20}$	$94.36_{\pm 0.16}$	$81.73_{\pm 0.25}$	$96.53_{\pm 0.19}$	$\mathbf{89.84}_{\pm 0.59}$	88.40	100.0
	BinSFO	$\mathbf{1/1}$	$92.61_{\pm 0.10}$	$\mathbf{75.35}_{\pm 0.32}$	$\mathbf{94.46}_{\pm 0.30}$	$\mathbf{82.15}_{\pm 0.43}$	$\mathbf{96.56}_{\pm 0.12}$	$89.59_{\pm 0.51}$	$\mathbf{88.45}$	$\mathbf{33.1}$
BCDNet	STE	32/1	$92.36_{\pm 0.24}$	$75.43_{\pm 0.20}$	$94.32_{\pm 0.39}$	$82.27_{\pm 0.32}$	$95.78_{\pm 0.31}$	$90.18_{\pm 0.21}$	88.39	73.5
	Bop	32/1	$92.29_{\pm 0.09}$	$75.45_{\pm 0.24}$	$94.36_{\pm 0.28}$	$82.45_{\pm 0.27}$	$95.83_{\pm 0.17}$	$90.05_{\pm 0.67}$	88.40	77.0
	ReSTE	32/1	$92.77_{\pm 0.22}$	$75.53_{\pm 0.07}$	$94.40_{\pm 0.28}$	$\mathbf{82.69}_{\pm 0.26}$	$96.04_{\pm 0.18}$	$90.22_{\pm 0.40}$	88.60	100.0
	BinSFO	$\mathbf{1/1}$	$\mathbf{92.85}_{\pm 0.14}$	$\mathbf{75.58}_{\pm 0.16}$	$\mathbf{94.44}_{\pm 0.24}$	$82.52_{\pm 0.23}$	$\mathbf{96.17}_{\pm 0.35}$	$\mathbf{90.54}_{\pm 0.72}$	$\mathbf{88.68}$	$\mathbf{29.3}$

tribution from the graphical models that represent the computational flows for SGD and BinSFO as shown in Figure 2. The first flow in Figure 2a binarizes parameters after applying the update rule of SGD in Eq. (1), which produces $\tilde{\boldsymbol{w}}_t = [\![\boldsymbol{w}_t \geq 0]\!] \in \mathbb{H}^N$. This is a conventional flow and requires that real-valued parameters be stored during training. The second flow in Figure 2b is the computational flow for BinSFO based on Eq. (6). Because $\boldsymbol{w}_{t-1} \in \mathbb{R}^N$ is not observable, it is treated as a latent variable. This flow eliminates the need to store real-valued parameters.

Given these two flows, the gap between SGD and BinSFO lies in the difference between $\tilde{\boldsymbol{w}}_t$ and $\boldsymbol{b}_t$. The oracle hypermask $\boldsymbol{m}_t^*$ that results in $\tilde{\boldsymbol{w}}_t = \boldsymbol{b}_t$ must exist. However, without observing $\boldsymbol{w}_{t-1}$, it is impossible to compute the oracle hypermask. To address this problem, we introduce a probabilistic approach and design the hypermask distribution that satisfies

$$\mathbb{E}[\tilde{\boldsymbol{w}}_t] = \mathbb{E}[\boldsymbol{b}_t], \tag{7}$$

where $\mathbb{E}[\cdot]$ denotes expected value.

To explicitly define the hypermask distribution, we assume that elements of $\boldsymbol{w}_{t-1}, \boldsymbol{g}_t$ are i.i.d. and follow Gaussian distributions; that is, we assume

$$g_{t,i} \sim N(0, \hat{\sigma}_t^2), \quad w_{t-1,i} \sim N(0, \tilde{\sigma}_{t-1}^2), \tag{8}$$

where $\hat{\sigma}_t$ is the variance of gradients, $\tilde{\sigma}_t^2 = \tilde{\sigma}_{t-1}^2 + \eta^2 \hat{\sigma}_t{}^2$ is the iteratively estimated parameter variance, and i is the element index. Under this assumption, the following hypermask distribution satisfies the condition of Eq. (7):

$$\begin{aligned} &P(m_{t,i} = 1 | w_{t-1,i}, g_{t,i}) \\ &= \begin{cases} \operatorname{erf}(\max(\tau_{t-1} g_{t,i}, 0)) & (w_{t-1,i} = 1) \\ -\operatorname{erf}(\min(\tau_{t-1} g_{t,i}, 0)) & (w_{t-1,i} = 0) \end{cases}, \end{aligned} \tag{9}$$

where $\tau_{t-1} = \eta/\sqrt{2}\sigma_{t-1}$ is a temperature and erf is the error function given by $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. This distribution was derived by applying Bayes' theorem to the joint distribution $P(\tilde{w}_{t,i}, w_{t-1,i}, g_{t,i})$ using the graphical model.

IV. Experiments

This section demonstrates the effectiveness of BinSFO in finetuning scenarios on six image classification datasets.

A. Experimental settings.

Datasets and evaluation metric. The six image classification datasets were used for finetuning: CIFAR-10 [27], CIFAR-100 [27], Caltech-101 [28], Caltech-256 [29], Flowers [30] and Pets [31]. Accuracy on each dataset was used as the primary evaluation metric. For pretraining, the ImageNet dataset [32] was used and a pretrained BNN was assumed to be given in the finetuning phase.

Network architectures. Two BNN architectures were used: ReActNet [20], [25] and BCDNet [23]. ReActNet is a convolutional architecture that uses the RPReLU activation. BCDNet is a hybrid architecture consisting of binary convolution blocks and binary MLP blocks.

Baselines. We chose three baselines: STE [8], Bop [26], and ReSTE [13]. They store real-valued parameters during finetuning, whereas BinSFO does not.

Implementation details. Each BNN was finetuned for 400 epochs using a cosine-decay scheduler with an initial learning rate of 10^{-2}. The SGD optimizer was used for the conventional methods. The variance $\hat{\sigma}_t$ was computed at each iteration from the observation of $\boldsymbol{g}_t$ and $\tilde{\sigma}_0$ was set to 1. ReActNet consisted of fourteen layers. BCDNet was constructed with nine convolutional blocks and eleven binary MLP blocks.

B. Experimental results

Full finetuning. Table I shows the full finetuning results. As shown, BinSFO performed comparable to the conventional methods, slightly outperforming ReSTE in terms of average accuracy across the six datasets, with memory consumption reduced by 66.9% and 70.7% with ReActNet and BCDNet, respectively. These results demonstrate both the effectiveness and the memory efficiency of BinSFO.

Partial finetuning. To analyze the trade-off between memory consumption and accuracy, we conducted partial finetuning experiments, where the parameters in the first k layers are frozen. Figure 3 shows the results. As shown, BinSFO was more memory-efficient and effective for all k, demonstrating the ability to significantly reduce memory consumption while maintaining or even improving accuracy.

C. Analysis

Comparison to the oracle hypermask. To evaluate the gap between the sampled hypermasks $\boldsymbol{m}_t$ and the oracle

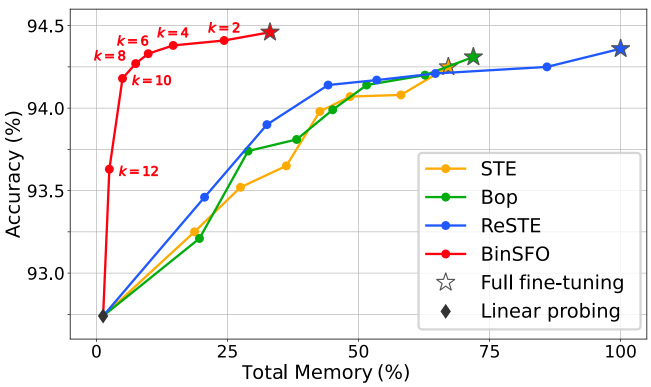


Fig. 3: Trade-off between memory consumption and accuracy. The number of frozen layers was varied as $k = 0, 2, 4, \cdots, 14$ for ReActNet [20] on Caltech-101.

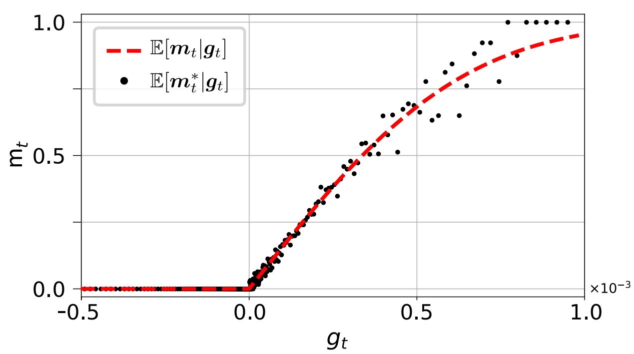


Fig. 4: Evaluation of hypermasks.

hypermasks $\boldsymbol{m}_t^*$ that satisfies $\tilde{\boldsymbol{w}}_t = \boldsymbol{b}_t$, Figure 4 plots the conditional expected values $\mathbb{E}[\boldsymbol{m}_t|\boldsymbol{g}_t]$ and $\mathbb{E}[\boldsymbol{m}_t^*|\boldsymbol{g}_t]$ as functions of $\boldsymbol{g}_t$. Note that black dots represent estimated values obtained through the flow in Figure 2a using real-valued parameters. As shown, the two expected values aligned well. This confirmed the correctness of the hypermask distribution in Eq. (9).

Justification of the assumption. To justify the assumption in Eq. (8), Figure 5 shows the distributions of the gradient values $\boldsymbol{g}_t$ and parameters $\boldsymbol{w}_{t-1}$. We observed that the gradient value distribution is very close to a Gaussian distribution with a mean of zero, but the parameter distribution gradually deviates from Gaussian as learning progresses. This shows that our assumption is reasonable, but there will likely be a need to relax the assumption in future work. Figure 6 shows scatterplots for the joint distribution. We observed no strong correlation between elements of $\boldsymbol{g}_t$ and $\boldsymbol{w}_{t-1}$, supporting our assumption that they are independent.

Memory. Figure 7 shows a breakdown of memory consumption during training. Since BinSFO eliminates the need to store real-valued parameters, it exhibits a significantly reduced memory consumption for storing the parameters. To further reduce the memory consumption, binarizing the whole backpropagation process would be challenging but interesting future work.

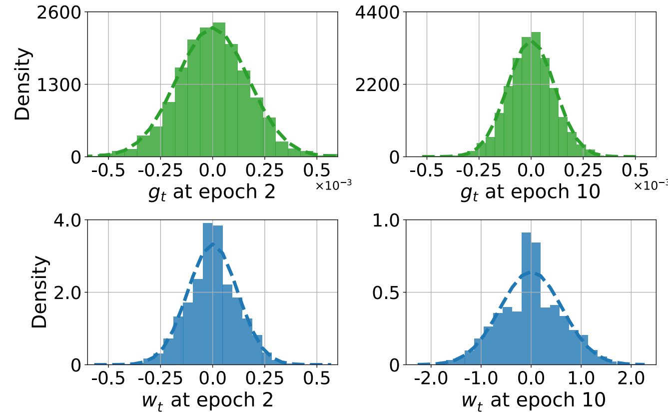


Fig. 5: Distributions of elements of $\boldsymbol{g}_t$ and $\boldsymbol{w}_t$ of the output layer. The dashed curves are zero-mean Gaussian distributions with the empirical variance.

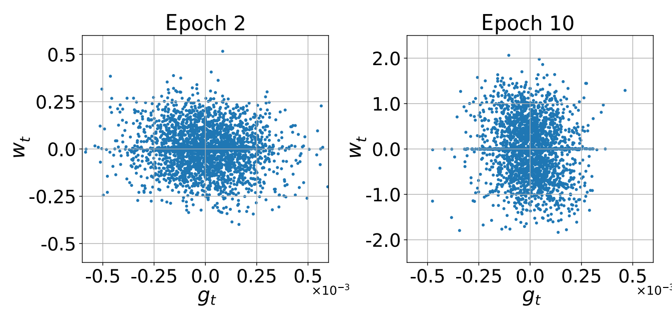


Fig. 6: Joint distribution of $\boldsymbol{g}_t$ and $\boldsymbol{w}_t$.

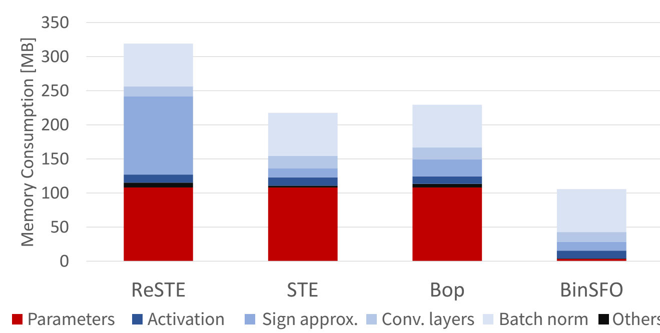


Fig. 7: Breakdown of the memory consumption for each component of ReActNet.

V. Conclusion and Discussion

This paper introduced BinSFO, a training algorithm for BNNs that eliminates the need to store real-valued parameters during training. We analyzed graphical models representing computational flows of SGD and BinSFO, and derived a probabilistic distribution from which a hypermask is sampled at each training iteration. In experiments, we demonstrated the effectiveness and memory efficiency of BinSFO on six image classification datasets. It was found that BinSFO performed comparable to ReSTE while consuming significantly less memory. For future work, fully binarizing the backpropagation procedure would seem to be an interesting direction to explore.

References

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[2] H. Kim, J. Shin, W. Kim, and A. A. D. Barrio, "1-d spatial attention in binarized convolutional neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[3] J. Tian, C. Fang, H. Wang, and Z. Wang, "Bebert: Efficient and robust binary ensemble bert," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[4] J. Zhang, Z. Su, Y. Feng, X. Lu, M. Pietikäinen, and L. Liu, "Dynamic binary neural network by learning channel-wise thresholds," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[5] M. Shen, X. Liu, R. Gong, and K. Han, "Balanced binary neural networks with gated residual," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[6] H. Zhao, Y. Xiao, J. Han, and Z. Zhang, "Compact convolutional recurrent neural networks via binarization for speech emotion recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[7] C. Sakr, J. Choi, Z. Wang, K. Gopalakrishnan, and N. Shanbhag, "True gradient-based training of deep binary activated neural networks via continuous binarization," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[8] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[9] A. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, "Forward and backward information retention for accurate binary neural networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2250–2259.

[10] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[11] Y. Xu, K. Han, C. Xu, Y. Tang, C. Xu, and Y. Wang, "Learning frequency domain approximation for binary neural networks," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[12] M. Lin, R. Ji, Z. Xu, B. Zhang, Y. Wang, Y. Wu, F. Huang, and C.-W. Lin, "Rotated binary neural network," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[13] X.-M. Wu, D. Zheng, Z. Liu, and W.-S. Zheng, "Estimator meets equilibrium perspective: A rectified straight through estimator for binary neural networks training," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[14] R. T. Icarte, L. Illanes, M. P. Castro, A. A. Cire, S. A. McIlraith, and J. C. Beck, "Training binarized neural networks using mip and cp," in *International Conference on Principles and Practice of Constraint Programming*, 2019.

[15] E. B. Khalil, A. Gupta, and B. Dilkina, "Combinatorial attacks on binarized neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[16] J. Kurtz and B. Bah, "Efficient and robust mixed-integer optimization methods for training binarized deep neural networks," *arXiv preprint arXiv:2110.11382*, 2021.

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. European Conference on Computer Vision (ECCV)*, 2016, pp. 525–542.

[18] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 345–353.

[19] 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 *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 722–737.

[20] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," in *ECCV*, 2020.

[21] P.-H. C. Le and X. Li, "Binaryvit: Pushing binary vision transformers towards convolutional models," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 4665–4674.

[22] Z. Liu, B. Oguz, A. Pappu, L. Xiao, S. Yih, M. Li, R. Krishnamoorthi, and Y. Mehdad, "Bit: Robustly binarized multi-distilled transformer," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[23] X. Xing, Y. Li, W. Li, W. Ding, Y. Jiang, Y. Wang, J. Shao, C. Liu, and X. Liu, "Towards accurate binary neural networks via modeling contextual dependencies," in *ECCV*, 2022.

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

[25] Z. Liu, Z. Shen, S. Li, K. Helwegen, D. Huang, and K.-T. Cheng, "How do adam and training strategies help bnns optimization?" in *Proc. International Conference on Machine Learning (ICML)*, 2021.

[26] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization," in *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[28] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[29] G. Gregory, H. Alex, and P. Pietro, *Caltech-256 Object Category Dataset*, 2007.

[30] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *The Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[31] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.