SURGE: SURROGATE GRADIENT ADAPTATION IN BINARY NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The training of Binary Neural Networks (BNNs) is fundamentally based on gradient approximation for non-differentiable binarization operations (*e.g.*, `sign` function). However, prevailing methods including the Straight-Through Estimator (STE) and its improved variants, rely on hand-crafted designs that suffer from gradient mismatch problem and information loss induced by fixed-range gradient clipping. To address this, we propose SURrogate GradiEnt Adaptation (SURGE), a novel learnable gradient compensation framework with theoretical grounding. SURGE mitigates gradient mismatch through auxiliary backpropagation. Specifically, we design a Dual-Path Gradient Compensator (DPGC) that constructs a parallel full-precision auxiliary branch for each binarized layer, decoupling gradient flow via output decomposition during backpropagation. DPGC enables bias-reduced gradient estimation by leveraging the full-precision branch to estimate components beyond STE's first-order approximation. To further enhance training stability, we introduce an Adaptive Gradient Scaler (AGS) based on an optimal scale factor to dynamically balance inter-branch gradient contributions via norm-based scaling. Experiments on image classification, object detection, and language understanding tasks demonstrate that SURGE performs best over state-of-the-art methods.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable success across various domains (He et al., 2016; Vaswani, 2017), with model parameters scaling from millions to billions in state-of-the-art architectures (Brown et al., 2020; Yang et al., 2024). However, their escalating computational complexity and memory requirements pose significant challenges for deployment in resource-limited scenarios. To address this challenge, numerous model compression techniques have been developed to enhance deployment efficiency (He & Xiao, 2023; Hinton et al., 2014; Liu et al., 2025; Yu et al., 2017), each offering distinct trade-offs among compression ratio, inference speedup, and accuracy retention. Different from structural compression methods (*e.g.*, pruning), quantization (Esser et al., 2019; Hubara et al., 2021; Wang et al., 2022; Xu et al., 2023) achieves compression through bit-width reduction without modifying the network architecture. The reduced bit-width representation significantly decreases storage requirements while enabling computational acceleration via low-precision operations.

As an extreme form of quantization, binarization (Courbariaux et al., 2015; 2016; Gong et al., 2019; Xu et al., 2021b; 2022a) represents weights and activations with 1-bit values, theoretically enabling $32\times$ memory reduction and $58\times$ computational acceleration compared to full-precision networks (Rastegari et al., 2016). These efficiency advantages of binarization make it especially practical for edge computing devices with severely limited computational resources, and its effectiveness has been proven in diverse tasks, such as classification (Xu et al., 2021c), object detection (Xu et al., 2022b), and natural language understanding (Qin et al., 2022).

Despite considerable advances, there remains a non-negligible performance gap between binary neural networks (BNNs) and their full-precision counterparts (Rastegari et al., 2016). This discrepancy primarily stems from the substantial representation divergence between binary and continuous-valued weights and activations. Specifically, the training of BNNs incorporates quantization of real-valued tensors with deterministic or stochastic binarization operations (Courbariaux et al., 2016).

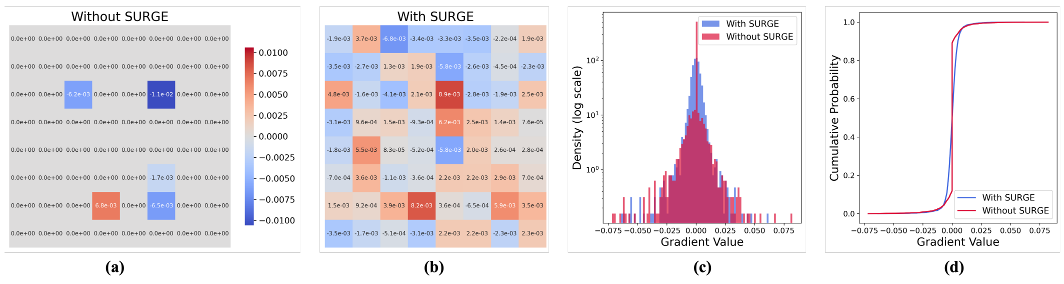

Figure 1: (a-b) Activation gradient patterns without/with SURGE (left/right); (c) Gradient distribution comparison; (d) Cumulative probability of gradients. STE provides a first-order approximation for the `sign` function's gradient and clips out-of-range activation gradients, while SURGE compensates them with a Dual-Path Gradient Compensator (a-b). SURGE also right-shifts gradient distributions of activations (c-d), validating its effectiveness in rectifying STE-induced mismatch.

However, the non-differentiable nature and vanishing gradients of binarization operations introduce significant challenges in backpropagation.

To solve the training problem, the Straight-Through Estimator (STE) (Bengio et al., 2013) provides an effective gradient approximation method for binarization operations. Specifically, STE directly substitutes the gradients of binarization operations (*e.g.*, `sign` function) with the derivative of the `Identity` function during backpropagation, thereby enabling stable parameter optimization. Despite its prevalent application in training BNNs and low-bit networks, STE suffers from several inherent limitations that remain to be addressed. On the one hand, since the `sign` function's gradient vanishes everywhere except at zero, employing a fixed-value gradient approximation inevitably introduces estimation bias and optimization instability (Qin et al., 2020). To reduce the gradient error of STE, subsequent approaches predominantly rely on heuristic quantizer designs (Liu et al., 2019; Gong et al., 2019), such as piecewise polynomial functions (Liu et al., 2018b) and SignSwish activation functions (Darabi et al., 2018), which cannot guarantee finding the optimal gradient approximation.

On the other hand, during the backpropagation of STE, the gradient clipping is adopted to only preserve the gradient for inputs within the vicinity of zero (typically $[-1, 1]$), which empirically improves model accuracy (Courbariaux et al., 2016). However, applying fixed-range gradient clipping is suboptimal for binarized representations, particularly for activation quantization, since the gradient information is discarded for values outside the clipping range (Qin et al., 2020). Existing binarization methods largely overlook the impact of gradient clipping range, as only a few studies propose handcrafted asymptotic functions to gradually approximate the hard binarization function (Gong et al., 2019; Qin et al., 2020). Consequently, merely employing STE and improved estimators (Rastegari et al., 2016; Gong et al., 2019; Xu et al., 2022a; Jin et al., 2025) fails to obtain accurate gradient approximation for binarization operations, as non-negligible gradient mismatch (Qin et al., 2020) accumulates in the backward pass, necessitating explicit gradient rectification.

This paper proposes SURrogate GradiEnt Adaptation (SURGE), a novel learnable gradient compensation strategy that addresses gradient mismatch through auxiliary backpropagation. While STE or improved estimators provides surrogate gradients for binarization operations, SURGE offers enhanced gradient adaptation for binary neural networks. Specifically, we design a Dual-Path Gradient Compensator (DPGC), which constructs a parallel full-precision parameterized branch (noted as auxiliary branch) for each binarized layer (noted as main branch). In particular, DPGC decomposes each layer's output into contributions from the main branch and auxiliary branch, thus decoupling the gradient flow into two parts during backpropagation. Therefore, DPGC ensures that the auxiliary branch only affects the backward gradient while preserving the original layer outputs during the forward pass. Compared with the binary branch, the full-precision branch can provide less biased gradients (Stock et al., 2021) that compensate for STE's first-order approximation (Liu et al., 2023) error by learning higher-order terms. As shown in Fig. 1, **(a)** STE's fixed clipping *zeros vast area* of activation gradients; **(b)** with SURGE, the auxiliary branch injects compensation gradients while keeping the forward output unchanged, visibly recovering the clipped regions. Aggregated statistics in **(c)**–**(d)** show a right-shifted gradient distribution and heavier tails in the cumulative curves, indicating that SURGE restores informative gradients beyond STE's first-order surrogate.

Moreover, large-magnitude gradients from the auxiliary path may adversely affect the convergence of the main branch. To address this problem, we propose an Adaptive Gradient Scaler (AGS) that dynamically balances inter-branch gradient contributions via norm-based scaling, thereby ensuring stable and effective compensation. To validate the effectiveness of SURGE, we conduct comprehensive comparative experiments on two image classification benchmarks, one object detection benchmark, one suite of language understanding benchmark, and our proposed method achieves best performance over state-of-the-art. In summary, the main contributions of this work are as follows:

- We propose SURrogate GradiEnt Adaptation (SURGE), a novel gradient compensation framework employing a Dual-Path Gradient Compensator to address gradient mismatch. Our method does not modify the forward-pass output and introduces no additional overhead at inference.
- We introduce an Adaptive Gradient Scaler (AGS) that dynamically equilibrates gradient contributions from binary and auxiliary branches based on theoretically derived optimal scaling factor.
- Extensive experiments demonstrate that SURGE achieves state-of-the-art performance across four standard benchmarks for BNN training. Specifically, a SURGE-trained binarized ResNet-18 attains 62.0% top-1 accuracy on ImageNet with one-stage training, surpassing previous SOTA methods by significant margins (*e.g.*, +1.0%, and +3.9% top-1 accuracy improvements over ReCU and IR-Net, respectively, on ImageNet).

2 Related Work

2.1 Gradient Approximation

Gradient approximation serves as a cornerstone for training neural networks with non-differentiable operators, addressing challenges in discrete sampling (Sutton et al., 1999; Schulman et al., 2015; Athalye et al., 2018; Rezende et al., 2014), architecture search (Xie et al., 2018; Liu et al., 2018a; Cai et al., 2018), and especially quantization (Esser et al., 2020; Gong et al., 2019; Liu et al., 2018b; 2020; Xu et al., 2022a). A popular family of gradient estimators is the Straight-Through Estimator (STE), which directly propagates gradients through non-differentiable functions. The idea of Straight-Through originates from the perceptron algorithm (Rosenblatt, 1957), which leverages a modified chain rule and utilizes the `Identity` function as the proxy of the original derivative of a binary output function. (Bengio et al., 2013) improves this method by using non-linear functions like sigmoid, and (Jang et al., 2016) further incorporates the Gumbel reparameterization, reparameterizes discrete variables via temperature-annealed continuous relaxation, enabling low-variance gradient estimation for categorical sampling. In the field of quantization, DSQ (Gong et al., 2019) employs parameterized sigmoid functions to progressively approximate the gradients of the non-differentiable quantization function, while LSQ (Esser et al., 2020) introduced scaling factors for end-to-end gradient propagation, advancing low-bit quantization. BONN (Zhao et al., 2022) integrates Bayesian optimization to guide differentiable binarization policies, and FDA-BNN (Xu et al., 2021b) converts the `sign` function into the frequency domain to mitigate the gradient mismatch.

2.2 Binary Neural Network

Pioneering works in binary neural networks focused either on binarization architecture design (Liu et al., 2018b; Xu et al., 2021b; Liu et al., 2020; Bulat et al., 2020; Yang et al., 2020) or training strategies (Courbariaux et al., 2015; Rastegari et al., 2016; Qin et al., 2020; Xu et al., 2021c; 2022a). In terms of architecture design, Bi-Real Net (Liu et al., 2018b) enhances skip connections, and FDA-BNN (Xu et al., 2021b) introduces differentiable binarization units in the frequency domain. Moreover, ReActNet (Liu et al., 2020) substitutes the `sign` function and PReLU (He et al., 2015) with RSign and RPReLU based on learnable thresholds. Approaches like BATS (Bulat et al., 2020) and SLB (Yang et al., 2020) combine BNNs with neural architecture search. In terms of training strategies, BinaryConnect (Courbariaux et al., 2015) and XNOR-Net (Rastegari et al., 2016) use the `sign` function with gradient approximation, but they cause severe information loss in forward propagation. Later, training strategies were innovated. IR-Net (Qin et al., 2020) and ReCU (Xu et al., 2021c) use progressive quantization and feature distribution alignment, but they still face gradient mismatch in deep networks. RBONN (Xu et al., 2022a) introduces a recurrent bilinear optimization for BNNs.

Unlike prior work, our work is the first attempt to employ a Dual-Path Gradient Compensator to correct gradient mismatch in STE-based binarized networks, coupled with an Adaptive Gradient Scaler to equilibrate the gradient contribution between binary and auxiliary branches dynamically.

3 Preliminaries

Consider a neural layer with weight vector $W \in \mathbb{R}^n$ and input vector $x \in \mathbb{R}^n$. The main operation in deep neural networks is expressed as:

$$f(x; W) = W^\top x. \tag{1}$$

In binary neural networks (BNNs), we quantize W and x to $\{-1, +1\}^n$, thus using the efficient XNOR and Bit-count operations to replace real-valued operations. Let $\mathbf{B}_W \in \{-1, +1\}^n$ and $\mathbf{B}_x \in \{-1, +1\}^n$ denote the binarized counterparts. Network binarization aims to represent the floating-point weights and/or activations with 1 bit. In general, the quantization can be formulated as: $Q_x(x) = \alpha_x \mathbf{B}_x, Q_W(W) = \alpha_W \mathbf{B}_W$, where $\alpha_.$ denotes scalars for binary values including α_w for weights and α_x for inputs. And we usually use `sign` function to binarize W and x: $\mathbf{B}_x = \mathtt{sign}(x), \mathbf{B}_W = \mathtt{sign}(W)$. Following Rastegari et al. (2016), the binary operation is formulated as:

$$f_b(x; \mathbf{B}_W) = Q_W(W)^\top Q_x(x) = \alpha_W \alpha_x \cdot (\mathbf{B}_W \odot \mathbf{B}_x), \tag{2}$$

where $\odot$ denotes the inner product for vectors with bitwise operations XNOR and Bitcount.

In backpropagation, the derivative of the `sign` function is zero almost everywhere, which makes it incompatible with backpropagation, since exact gradients for the original values before binarization would be zeroed. Thus, Straight-Through Estimator (STE) (Bengio et al., 2013) is generally used to train BNNs, which propagates the gradient through `Identity` function. Regarding the gradient of the loss L *w.r.t.* W, it is approximated as

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \mathbf{B}_W} \cdot \frac{\partial \mathbf{B}_W}{\partial W} \approx \frac{\partial L}{\partial \mathbf{B}_W} \tag{3}$$

As for the gradient *w.r.t.* the activations, it can be formulated as

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial \mathbf{B}_x} \cdot \frac{\partial \mathbf{B}_x}{\partial x} \approx \frac{\partial L}{\partial \mathbf{B}_x} \cdot \mathbf{1}_{\{|x| \leq 1\}} \tag{4}$$

where $\mathbf{1}_{\{|x| \leq 1\}}$ is the indicator function that equals 1 when $|x| \leq 1$ and 0 otherwise. This expression corresponds to STE's first-order approximation for the sign function's gradient.

4 Methodology

In this section, we describe SURGE in detail. We first introduce the Dual-Path Gradient Compensator (DPGC) module to address the gradient mismatch in STE-based training (Sec. 4.1), then present the Adaptive Gradient Scaler (AGS) for stable optimization (Sec. 4.2). The complete training paradigm integrates these components while preserving standard BNN inference.

4.1 Dual-Path Gradient Compensator (DPGC)

To handle the intrinsic gradient mismatch in STE (Qin et al., 2020), we propose a layer-wise dual-path architecture that preserves original forward computations while introducing auxiliary gradient pathways. As shown in Fig. 2, DPGC constructs a parallel full-precision parameterized branch (noted as auxiliary branch) including a full-precision operator (*e.g.*, convolution, linear, attention

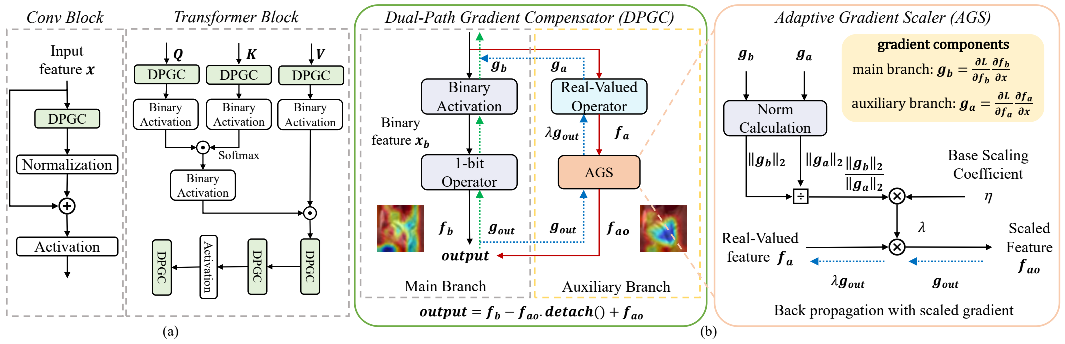

Figure 2: Overall architecture of SURGE. **(a)** Integration into common backbones (left: convolution block; right: transformer block). **(b)** Component details. DPGC constructs a parallel full-precision parameterized branch (auxiliary branch, shown with red arrows for forward pass and blue arrows for backpropagation) for each binarized layer (main branch, represented by black arrows in forward pass and green arrows for backpropagation). This ensures identical output to standard BNNs while providing less biased gradients for compensation. AGS takes gradients from both branches as input (visualized through corresponding colored arrows) and dynamically balances inter-branch gradient contributions via norm-based scaling. SURGE is architecture-agnostic and applies to arbitrary binarized linear operators.

projection) with identical dimensions (*e.g.*, kernel size, dimension) to the main branch, augmented with an Adaptive Gradient Scaler (AGS) module (Section 4.2) for each binarized layer (noted as main branch). DPGC decomposes each layer's output into contributions from both the main branch (black arrow) and auxiliary branch (red arrow), thus decoupling the gradient flow into two parts during backpropagation (Eq. 6) (green arrow for main branch, blue arrow for auxiliary branch). Given $f_b(x) = Q_W(W_b)^\top Q_x(x)$ as the binarized computation, $f_a(x) = W_a^\top x$ as the full-precision computation, W_a, W_b as the weight parameters for auxiliary branch (full-precision branch), main branch (binary branch), respectively, the combined output is:

$$\mathit{output} = \underbrace{f_b(x;W_b)}_{\text{Binary output}} - \underbrace{f_{ao}(x;W_a)\downarrow}_{\text{Detached compensator}} + \underbrace{f_{ao}(x;W_a)}_{\text{Active compensator}}, \tag{5}$$

where $f_{ao}(x) = \lambda f_a(x)$ is the scaled full-precision computation, λ is the scale factor, $\downarrow$ is the gradient stop operator. The design ensures identical outputs to standard BNNs, while gradients flow through both pathways, thus providing less biased gradient estimates (Stock et al., 2021) while preserving STE gradients. Upon completion of training, the auxiliary branch can be discarded, introducing no additional computational overhead during inference. Backpropagation aggregates gradients from both paths:

$$\frac{\partial\mathcal{L}}{\partial x} = \underbrace{\frac{\partial\mathcal{L}}{\partial f_b}\frac{\partial f_b}{\partial x}\Big|_{\text{STE}}}_{\text{Binary gradients } g_b} + \lambda \underbrace{\frac{\partial\mathcal{L}}{\partial f_a}\frac{\partial f_a}{\partial x}}_{\text{Compensator gradients } g_a}. \tag{6}$$

Here, g_b retains STE's first-order approximation (Liu et al., 2023), while g_a from the full precision auxiliary branch captures higher-order terms and retains more gradient information excluded by clipping.

4.2 Adaptive Gradient Scaler (AGS)

The raw combination of g_b and g_a risks unstable training due to varying magnitude ratios between paths, and the large-magnitude gradients from the auxiliary path may adversely affect the convergence of the main branch. We address this through a novel mechanism that dynamically balances inter-branch gradient contributions with norm-based adaptive scaling factor λ^*, thereby ensuring stable and effective compensation:

$$\tilde{g} = g_b + \lambda g_a, \quad \lambda^* = \eta \frac{\|g_b\|_2}{\|g_a\|_2 + \epsilon}, \tag{7}$$

where η is the base scaling coefficient, $\epsilon = 10^{-8}$ is the numerical stabilizer. This dynamic scaling preserves the directional consistency of the primary binary gradient g_b while allowing auxiliary gradients g_a to provide magnitude-aware compensation. Such design guarantees that the STE-based gradients dominate the parameter update process, while the auxiliary path serves as an adaptive compensator that injects higher-order gradient information without destabilizing the primary learning dynamics. In practice, the scale factor derived from gradient computation in the current iteration is used in the subsequent AGS step for adaptive parameter adjustment to optimize computational efficiency. The complete training procedure is summarized in Appendix A.

5 Theoretical Analysis

This section formally establishes the theoretical foundation of gradient compensation in dual-path architectures. We begin by formulating the gradient propagation mechanism under our proposed compensation framework, followed by rigorous definitions of gradient components to characterize their statistical properties (Definition 1). Finally, we derive the theoretically optimal scaling factor for gradient compensation and demonstrate its alignment with our adaptive scaling strategy (Theorem 1).

Let $\mathcal{X}$ denote the input space and $W = (W_b, W_a) \in \mathbb{R}^{2d}$ represent the binarized and full-precision weights. The forward propagation becomes:

$$f(x; W) = \underbrace{Q_W(W_b)^\top Q_x(x)}_{\text{Binary path}} + \lambda \left(\underbrace{W_a^\top x}_{\text{Compensator path}} - \underbrace{W_a^\top x \downarrow}_{\text{Detached path}} \right), \tag{8}$$

where λ follows the adaptive scaling in Section 4.2. Let $Approx$ denote a kind of STE-based gradient approximation (*e.g.*, STE), the composite gradient combines:

$$\frac{\partial \mathcal{L}}{\partial x} = \underbrace{\frac{\partial \mathcal{L}}{\partial f_b} \frac{\partial f_b}{\partial x}\Big|_{Approx}}_{g_b} + \underbrace{\frac{\partial \mathcal{L}}{\partial f_{ao}} \frac{\partial f_{ao}}{\partial x}}_{\lambda g_a}. \tag{9}$$

Definition 1 (Gradient Components)**.** *Assuming there exists g^* being a better gradient, the empirical gradients satisfy:*

$$\mathbb{E}[g_b] = g^* - \delta_b, \quad \|\delta_b\| \le C\sqrt{d}, \tag{10}$$

$$\mathrm{Var}(g_b) = \sigma_b^2 I_d, \quad \mathrm{Var}(g_a) = \sigma_a^2 I_d, \tag{11}$$

where C is a constant, which depends on the activation distribution. $\mathbb{E}[\cdot]$ and $\mathrm{Var}(\cdot)$ denote expectation and variance. δ_b represents the $Approx$-induced error vector, σ_b and σ_a are the standard deviations of gradient noise in binarized and compensator paths, respectively. I_d is the d-dimensional identity matrix. The full list of assumptions underlying this definition is provided in Appendix B.1.

Theorem 1 (Optimal Scaling Factor)**.** *Given numerical stabilizer $\epsilon = 10^{-8}$, the optimal scaling factor λ^*, minimizing total error $\mathbb{E}[\|\tilde{g} - g^*\|^2]$, can be approximated by multiplying a small constant η with the fraction $\frac{\|g_b\|}{\|g_a\|}$ as below:*

$$\lambda^* = \frac{\langle \delta_b, \mathbb{E}[g_a] \rangle}{\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2} \approx \eta \frac{\|g_b\|}{\|g_a\| + \epsilon}. \tag{12}$$

The approximation replaces expectations with empirical mini-batch estimates.

The proof is detailed in Appendix B.2. We now derive the expression for the optimal scaling factor λ^*, which is norm-based and adaptive, adopted in our AGS module (Sec. 4.2). This analysis establishes a principled gradient scaling factor that improves the resulting gradient update and alleviates the gradient mismatch.

Table 1: Performance comparison with the state-of-the-arts on CIFAR-10. W/A denotes the bit length of the weights and activations.

Network	Method	W/A	Top-1
ResNet-18	Real-Valued	32/32	94.8%
	RAD	1/1	90.5%
	IR-Net	1/1	91.5%
	RBNN	1/1	92.2%
	ReCU	1/1	92.8%
	SURGE (Ours)	1/1	**93.1%**
ResNet-20	Real-Valued	32/32	92.1%
	DoReFa	1/1	79.3%
	DSQ	1/1	84.1%
	SLB	1/1	85.5%
	IR-Net	1/1	86.5%
	ReCU	1/1	87.4%
	SURGE (Ours)	1/1	**88.0%**
VGG-Small	Real-Valued	32/32	94.1%
	XNOR-Net	1/1	89.8%
	DoReFa	1/1	90.2%
	IR-Net	1/1	90.4%
	RBNN	1/1	91.3%
	DSQ	1/1	91.7%
	SLB	1/1	92.0%
	ReCU	1/1	92.2%
	SURGE (Ours)	1/1	**92.5%**

6 Experiments

6.1 Datasets and Implementation Details

Datasets. We evaluate on two standard image classification benchmarks, one object detection benchmark, and one suite of language understanding tasks to demonstrate the effectiveness: CIFAR-10 (Krizhevsky et al., 2009), ImageNet-1K (Russakovsky et al., 2015), PASCAL VOC (Everingham et al., 2010), and GLUE (Wang et al., 2018). More details of datasets, data augmentation, and evaluating metrics are provided in Appendix D.

Implementation Details. On CIFAR-10, we evaluate our method with ResNet-18/20 (He et al., 2016) and VGG-Small (Simonyan & Zisserman, 2014). On PASCAL VOC, we binarize Faster-RCNN (Ren et al., 2016) with a ResNet-18 backbone. On GLUE, we evaluate our method with BERT-base (Devlin et al., 2019). More training details are provided in Appendix E.

6.2 Image Classification

CIFAR-10. We first show the experimental results on CIFAR-10 with ResNet-18, ResNet-20, VGG-Small backbone in Tab. 1. Specifically, we compare SURGE with state-of-the-art methods include RAD (Ding et al., 2019), IR-Net (Qin et al., 2020), RBNN (Lin et al., 2020), ReCU (Xu et al., 2021c), DoReFa (Zhou et al., 2016), DSQ (Gong et al., 2019), SLB (Yang et al., 2020), IR-Net (Qin et al., 2020), and XNOR-Net (Rastegari et al., 2016). We can see that SURGE outperforms all other methods in all backbones. Compared to recent ReCU, SURGE obtains a 0.3% performance increase with ResNet-18, a 0.6% performance increase with ResNet-20, and a 0.3% performance increase with VGG-Small.

One-Stage Training on ImageNet. Tab. 2 displays the performance comparison in binarizing ResNet-18 with one-stage training on ImageNet. We compare SURGE with DoReFa (Zhou et al., 2016), TBN (Wan et al., 2018), BNN (Courbariaux et al., 2016), XNOR-Net (Rastegari et al., 2016), Bi-Real Net (Liu et al., 2018b), IR-Net (Qin et al., 2020), BONN (Zhao et al., 2022), RBNN (Lin et al., 2020), RBONN (Xu et al., 2022a). We can see that SURGE is leading in both the top-1 and top-5 accuracies. Specifically, SURGE outperforms RBONN by 0.6% in top-1 accuracy, achieving the best performance.

Two-Stage Training on ImageNet. Tab. 3 displays the performance comparison in binarizing ResNet-18 with two-stage training on ImageNet. We compare SURGE with ReActNet (Liu et al., 2020), ReCU (Xu et al., 2021c), and RBONN (Xu et al., 2022a). Results show that SURGE outperforms all other methods in top-1 accuracy. Specifically, SURGE obtains a 0.2% performance increase in top-1 over RBONN. Our SURGE demonstrates superior overall performance compared to all existing approaches.

Table 2: A performance comparison with SOTAs on ImageNet with one-stage training. W/A denotes the bit length of weights and activations. We report the Top-1 (%) and Top-5 (%) accuracy performances.

Network	Method	W/A	OPs ($\times 10^8$)	Top-1	Top-5
ResNet-18	Real-valued	32/32	18.19	69.6	89.2
	DoReFa	1/4	2.44	59.2	81.5
	TBN	1/2	1.81	55.6	79.0
	BNN	1/1	1.63	42.2	67.1
	XNOR-Net			51.2	73.2
	Bi-Real Net			56.4	79.5
	IR-Net			58.1	80.0
	BONN			59.3	81.6
	RBNN			59.6	81.6
	ReCU			61.0	82.6
	RBONN			61.4	83.5
	SURGE (Ours)			**62.0**	**83.7**

Table 3: A performance comparison with SOTAs on ImageNet with two-stage training. W/A denotes the bit length of weights and activations. We report the Top-1 (%) and Top-5 (%) accuracy performances. * denotes the result is from the official checkpoint.

Network	Method	W/A	OPs ($\times 10^8$)	Top-1	Top-5
ResNet-18	Real-valued	32/32	18.19	69.6	89.2
	ReActNet	1/1	1.63	65.9	-
	ReCU			66.4	86.5
	RBONN*			66.5	**86.7**
	SURGE (Ours)			**66.7**	**86.7**

6.3 Object Detection

On the PASCAL VOC dataset, we compare the proposed SURGE against existing state-of-the-art binarized detection methods, such as ReActNet (Liu et al., 2020), LWS-Det (Xu et al., 2021a), and IDa-Det (Xu et al., 2022b), on the Faster-RCNN framework for object detection. The detection result of multi-bit quantized networks DoReFa-Net (Zhou et al., 2016) is also reported. As shown in Tab. 4, compared with the prior state-of-the-art IDa-Det, our method gains 0.5% performance increase, with the same FLOPs and memory usage. Compared with the raw real-valued detectors, SURGE surpasses real-valued Faster-RCNN with ResNet-18 backbone (77.0% $v.s.$ 76.4%) by apparent computation acceleration and storage savings by $5.21\times$ and $6.80\times$.

Table 4: Performance comparison of different methods in Faster-RCNN framework with input resolution set to 1000×600. † denotes that the result is from our re-implementation.

Framework	Backbone	Method	W/A	Memory Usage (MB)	OPs ($\times 10^9$)	mAP
Faster-RCNN	ResNet-18	Real-valued	32/32	112.88	96.40	78.8
		DoReFa-Net	4/4	21.59	27.15	73.3
		ReActNet	1/1	16.61	18.49	69.6
		LWS-Det				73.2
		IDa-Det†				76.5
		SURGE (Ours)				**77.0**

6.4 Language Understanding

On the GLUE dataset, we compare the proposed SURGE against existing state-of-the-art methods, such as BinaryBERT (Bai et al., 2020), BiBERT (Qin et al., 2022), and BiT (Liu et al., 2022), on BERT. We can see that SURGE outperforms all other methods. Specifically, SURGE obtains a 1.4% performance increase compared to BiT, and outperforms BiBERT by 8.9%, achieving the best performance.

Table 5: Performance comparison of BERT quantization on the GLUE dev set. FP is short for full precision. † denotes our re-implementation without multi-distillation techniques for fair comparison.

Quant	Size (MB)	FLOPs (G)	MNLI$_{m/mm}$	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
BERT (FP)	418	22.5	84.9/85.5	91.4	92.1	93.2	59.7	90.1	86.3	72.2	83.9
BinaryBERT	16.5	0.4	35.6/35.3	66.2	51.5	53.2	0	6.1	68.3	52.7	41.0
BiBERT	13.4	0.4	66.1/67.5	84.8	72.6	88.7	25.4	33.6	72.5	57.4	63.2
BiT†	13.4	0.4	77.0/77.5	85.4	85.5	87.8	23.6	68.0	79.4	58.1	70.6
SURGE(Ours)	13.4	0.4	77.3/77.5	87.1	86.2	88.6	24.1	71.7	80.6	60.6	**72.0**

6.5 Ablation Study

Ablation on Components. We ablate each component on CIFAR-10 using ResNet20. As shown in Tab. 6a, the baseline achieves 87.4% accuracy. Introducing the **Dual-Path Gradient Compensator (DPGC)** alone improves performance by +0.4%, validating its capability to balance gradient conflicts. Subsequent integration of the **Adaptive Gradient Scaler (AGS)** adds another +0.2%, demonstrating that AGS effectively modulates gradient magnitudes without disrupting DPGC's compensation. The hierarchical gains confirm that both mechanisms address distinct aspects of gradient optimization.

Table 6: Ablation Study on CIFAR-10 with ResNet20.

(a) Ablation on components

Method	Accuracy (%)
Baseline	87.4
+ DPGC	87.8
+ DPGC + AGS	**88.0**

(b) Ablation on gradient compensation scope

Method	Scope	Accuracy (%)
Baseline	/	87.4
SURGE	clipped gradients	87.7
	unclipped gradients	87.6
	all gradients	**88.0**

Ablation on Parameter η. As shown in Fig. 3, the performance degradation of fixed scaling ($scale > 0.05$, max -17.7%) highlights the necessity of dynamic adaptation, while our adaptive scaling achieves peak accuracy (87.95%) at $\eta = 0.01$. The result confirms that our theory-driven design (Theorem 1) successfully balances gradient compensation and training stability.

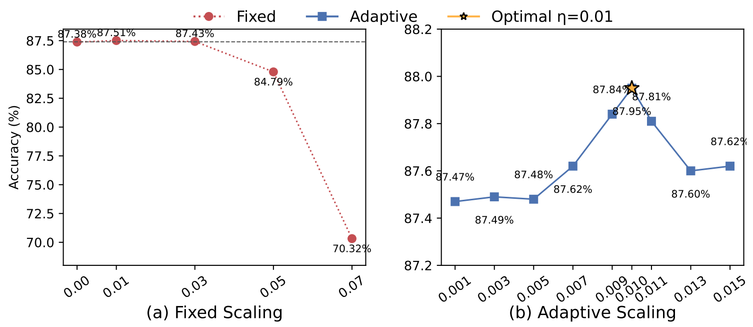

Figure 3: Ablation study on parameter scaling strategies. (a) is fixed scaling with constant factors across training iterations. (b) is adaptive scaling via parameter η that dynamically adjusts the compensation strength (Eq. 12).

Ablation on Gradient Compensation Scope of DPGC. We ablate the gradient compensation scope on CIFAR-10 using ResNet20. As detailed in Table 6b, compensating *only* gradients outside STE's clipping range ($|x| > 1$) yields 87.7% accuracy (0.3% improvement over baseline), while compensating *solely* within-range gradients ($|x| \leq 1$) achieves 87.6% (0.2% improvement). This verifies that both clipped and preserved gradient components contribute to parameter optimization. When jointly compensating *all* activation gradients through SURGE's adaptive integration, accuracy rises to 88.0% (0.6% improvement). This confirms that SURGE's design overcomes fixed-range clipping limitations in STE, enabling comprehensive gradient utilization.

7 Conclusion

This paper proposes a novel gradient compensation strategy that mitigates the STE-induced gradient mismatch through an auxiliary backpropagation. The proposed Dual-Path Gradient Compensator (DPGC) utilizes a dual-path architecture that ensures identical output to standard BNNs while providing less biased gradients for compensation. And the Adaptive Gradient Scaler (AGS) dynamically balances inter-branch gradient contributions via norm-based scaling. SURGE obtains the best performance over existing methods through main benchmarks in image classification, object detection, and language understanding tasks.

Ethics Statement

This work presents a novel gradient compensation framework (SURGE) for improving the training of Binary Neural Networks (BNNs). Our research is foundational and methodological in nature, focusing on algorithmic innovation. All experiments are conducted using standard, publicly available benchmark datasets (*e.g.*, CIFAR-10, ImageNet-1k, PASCAL VOC, and GLUE), and their use aligns with established academic practices. This work does not involve human subjects, personal data, or any form of social risk assessment.

We encourage the use of this technology for beneficial purposes and in accordance with ethical guidelines and legal standards. The authors declare no potential conflicts of interest.

Reproducibility Statement

To ensure the reproducibility of our work, we have provided comprehensive details necessary to replicate our results. The proposed SURGE algorithm, including the Dual-Path Gradient Compensator (DPGC) and Adaptive Gradient Scaler (AGS), is described in detail in Section 4. A complete description of our experimental setup, including network architectures, training hyperparameters, and optimization settings, is provided in Appendix E. Furthermore, we have provided our code in the supplementary materials, which contains the full implementation of our method and training scripts to facilitate easy replication and future research.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Adv. Neural Inform. Process. Syst.*, 33:1877–1901, 2020.

Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. Bats: Binary architecture search. In *Eur. Conf. Comput. Vis.*, pp. 309–325. Springer, 2020.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Adv. Neural Inform. Process. Syst.*, 28, 2015.

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.

Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia. Bnn+: Improved binary network training. 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ruizhou Ding, Ting-Wu Chin, Zeye Liu, and Diana Marculescu. Regularizing activation distribution for training binarized deep networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 11408–11417, 2019.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *Int. Conf. Learn. Represent.*, 2019.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *Int. Conf. Learn. Represent.*, 2020.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88:303–338, 2010.

Jianfei Feng. Bolt. `https://github.com/huawei-noah/bolt`, 2021.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *IEEE Int. Conf. Comput. Vis.*, pp. 4852–4861, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE Int. Conf. Comput. Vis.*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2016.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(5):2900–2919, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2014.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *Int. Conf. Mach. Learn.*, 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Lisa Jin, Jianhao Ma, Zechun Liu, Andrey Gromov, Aaron Defazio, and Lin Xiao. Parq: Piecewise-affine regularized quantization. *arXiv preprint arXiv:2503.15748*, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Adv. Neural Inform. Process. Syst.*, 33:7474–7485, 2020.

Boyu Liu, Haoyu Huang, Linlin Yang, Yanjing Li, Guodong Guo, Xianbin Cao, and Baochang Zhang. Efficient low-bit quantization with adaptive scales for multi-task co-training. In *Int. Conf. Learn. Represent.*, 2025.

Chunlei Liu, Wenrui Ding, Xin Xia, Baochang Zhang, Jiaxin Gu, Jianzhuang Liu, Rongrong Ji, and David Doermann. Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 2691–2699, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018a.

Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, and Jianfeng Gao. Bridging discrete and backpropagation: Straight-through and beyond. *Advances in Neural Information Processing Systems*, 36:12291–12311, 2023.

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 *Eur. Conf. Comput. Vis.*, pp. 722–737, 2018b.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Eur. Conf. Comput. Vis.*, pp. 143–159. Springer, 2020.

Zechun Liu, Barlas Oguz, Aasish Pappu, Lin Xiao, Scott Yih, Meng Li, Raghuraman Krishnamoorthi, and Yashar Mehdad. Bit: Robustly binarized multi-distilled transformer. *Advances in neural information processing systems*, 35:14303–14316, 2022.

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/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 2250–2259, 2020.

Haotong Qin, Yifu Ding, Mingyuan Zhang, YAN Qinghua, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. Bibert: Accurate fully binarized bert. In *International Conference on Learning Representations*, 2022.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Eur. Conf. Comput. Vis.*, pp. 525–542. Springer, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115:211–252, 2015.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. In *ICLR 2021-International Conference on Learning Representations*, 2021.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

A Vaswani. Attention is all you need. *Adv. Neural Inform. Process. Syst.*, 2017.

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 *Eur. Conf. Comput. Vis.*, pp. 315–332, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Longguang Wang, Xiaoyu Dong, Yingqian Wang, Li Liu, Wei An, and Yulan Guo. Learnable lookup table for neural network quantization. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2022.

Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 2049–2058, 2020.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Sheng Xu, Junhe Zhao, Jinhu Lu, Baochang Zhang, Shumin Han, and David Doermann. Layer-wise searching for 1-bit detectors. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 5682–5691, 2021a.

Sheng Xu, Yanjing Li, Tiancheng Wang, Teli Ma, Baochang Zhang, Peng Gao, Yu Qiao, Jinhu Lü, and Guodong Guo. Recurrent bilinear optimization for binary neural networks. In *Eur. Conf. Comput. Vis.*, pp. 19–35. Springer, 2022a.

Sheng Xu, Yanjing Li, Bohan Zeng, Teli Ma, Baochang Zhang, Xianbin Cao, Peng Gao, and Jinhu Lü. Ida-det: An information discrepancy-aware distillation for 1-bit detectors. In *Eur. Conf. Comput. Vis.*, pp. 346–361. Springer, 2022b.

Sheng Xu, Yanjing Li, Mingbao Lin, Peng Gao, Guodong Guo, Jinhu Lü, and Baochang Zhang. Q-detr: An efficient low-bit quantized detection transformer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2023.

Yixing Xu, Kai Han, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Learning frequency domain approximation for binary neural networks. *Adv. Neural Inform. Process. Syst.*, 34:25553–25565, 2021b.

Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. Recu: Reviving the dead weights in binary neural networks. In *IEEE Int. Conf. Comput. Vis.*, pp. 5198–5208, 2021c.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. *Adv. Neural Inform. Process. Syst.*, 33: 4091–4102, 2020.

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 7370–7379, 2017.

Junhe Zhao, Sheng Xu, Baochang Zhang, Jiaxin Gu, David Doermann, and Guodong Guo. Towards compact 1-bit cnns via bayesian learning. *Int. J. Comput. Vis.*, pp. 1–25, 2022.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

APPENDIX

A Training Procedure

The complete training procedure is summarized in Algorithm A.

Algorithm A Layer-wise Training with DPGC & AGS

```
Require: Layer input $x^{(l)}$, target $y$, learning rate $\alpha$, base scaling coefficient $\eta$, loss function $\mathcal{F}$
Ensure: Trained binarized weights $\{W_b^{(l)}\}_{l=1}^{L}$
 1: Initialize layer parameters $W_b^{(l)}, W_a^{(l)}$ {Binary & auxiliary full-precision paths}
 2: Initialize $\lambda^{(l,0)} \leftarrow \frac{1}{\sqrt{|W_a^{(l)}|}}$ {Reciprocal sqrt of auxiliary weight cardinality $|W_a^{(l)}|$}
 3: for iteration $t = 1$ to $T$ do
 4:     Forward Propagation (Layer $l$):
 5:     Compute binary path: $f_b^{(l)} \leftarrow W_b^{(l)} \odot \text{Sign}(x^{(l)})$
 6:     Compute auxiliary path: $f_a^{(l)} \leftarrow W_a^{(l)} \odot x^{(l)}$
 7:     Generate compensator: $f_{ao}^{(l)} \leftarrow \lambda^{(l,t-1)} \odot f_a^{(l)}$ {Previous scaling factor}
 8:     Synthesis output: $\text{out}^{(l)} \leftarrow f_b^{(l)} - f_{ao}^{(l)} \downarrow + f_{ao}^{(l)}$ {Forward synthesis via gradient-decoupled
        decomposition, detach gradient at $\downarrow$}
 9:     Loss Computation:
10:     Calculate Loss $\mathcal{L}$
11:     Backward Propagation (Layer $l$):
12:     Compute main branch gradients:
13:         $g_b \leftarrow \frac{\partial \mathcal{L}}{\partial f_b}\frac{\partial f_b}{\partial x}\Big|_{\text{STE}}, \quad g_{wb}^{(l)} \leftarrow \frac{\partial \mathcal{L}}{\partial W_b^{(l)}}\Big|_{\text{STE}}$
14:     Compute auxiliary branch gradients:
15:         $g_a \leftarrow \frac{\partial \mathcal{L}}{\partial f_a}\frac{\partial f_a}{\partial x}, \quad g_{wa}^{(l)} \leftarrow \lambda^{(l,t-1)} \odot \frac{\partial \mathcal{L}}{\partial W_a^{(l)}}$
16:     Then we have: $\frac{\partial \mathcal{L}}{\partial x^{(l)}} = g_b + \lambda g_a$
17:     Adaptive Gradient Scaler:
18:     Calculate norm ratio: $r \leftarrow \|g_b\|_2 / (\|g_a\|_2 + \epsilon)$
19:     Update scaling factor: $\lambda^{(l,t)} \leftarrow \eta \cdot r$
20:     Parameter Update (Layer $l$):
21:     $W_b^{(l)} \leftarrow W_b^{(l)} - \alpha \cdot g_{wb}^{(l)}$
22:     $W_a^{(l)} \leftarrow W_a^{(l)} - \alpha \cdot g_{wa}^{(l)}$
23: end for
```

B Theoretical Foundations and Proofs

B.1 Assumptions Underlying Definition 1

In this subsection we make explicit the assumptions underlying Definition 1 and Theorem 1. Intuitively, since the derivative of `sign` is zero almost everywhere and corresponds to a Dirac delta distribution at the origin in the sense of distributions, it is natural to view g^* as the gradient induced by an "ideal" surrogate that captures this behavior, while practical rules such as STE provide tractable but biased approximations.

Assumption 1 (Ideal reference gradient from a surrogate family)**.** *Consider a family $\mathcal{S}$ of smooth surrogate functions $s : \mathbb{R} \to \mathbb{R}$ that approximate the non-differentiable* `sign` *function used in binarization. For each $s \in \mathcal{S}$, let $\mathcal{L}_s(W)$ denote the population loss of the corresponding surrogate network. We assume that there exists a surrogate $s^* \in \mathcal{S}$ that attains the smallest population loss within this family, and define the associated reference ("better") gradient at the current parameter W as*

$$g^* := \nabla_W \mathcal{L}_{s^*}(W).$$

This g^* is not observable in practice and we never require a closed-form expression for it; it serves as an ideal target that practical surrogate gradients aim to approximate. We assume that g^* has finite second moments.

Assumption 2 (Empirical gradients as random vectors)**.** *At a fixed parameter W, the empirical gradients $g_b, g_a \in \mathbb{R}^d$ obtained from a single mini-batch are modelled as random vectors whose randomness comes from mini-batch sampling, data noise, and the stochastic optimization procedure. All expectations $\mathbb{E}[\cdot]$ and variances $\mathrm{Var}(\cdot)$ in our analysis are taken with respect to this randomness, and the empirical gradients have finite second moments.*

Assumption 3 (Bounded approximation-induced bias of STE)**.** *The baseline surrogate gradient g_b (e.g., using STE) is a biased approximation of the ideal reference gradient g^*:*

$$\mathbb{E}[g_b] = g^* - \delta_b,$$

where the approximation-induced bias $\delta_b \in \mathbb{R}^d$ is uniformly bounded in ℓ_2 norm:

$$\|\delta_b\| \le C\sqrt{d},$$

for some constant $C > 0$ that may depend on the activation distribution but not on the dimension d.

Assumption 4 (Isotropic, homoscedastic gradient noise)**.** *We decompose the empirical gradients as*

$$g_b = \mathbb{E}[g_b] + \varepsilon_b, \qquad g_a = \mathbb{E}[g_a] + \varepsilon_a,$$

where the noise terms satisfy $\mathbb{E}[\varepsilon_b] = \mathbb{E}[\varepsilon_a] = 0$. We assume an isotropic, homoscedastic noise model: there exist scalars $\sigma_b^2, \sigma_a^2 \ge 0$ such that

$$\mathrm{Var}(g_b) = \mathbb{E}[\varepsilon_b \varepsilon_b^\top] = \sigma_b^2 I_d, \qquad \mathrm{Var}(g_a) = \mathbb{E}[\varepsilon_a \varepsilon_a^\top] = \sigma_a^2 I_d,$$

where I_d is the d-dimensional identity matrix.

B.2 Proof of Theorem 1

Proof. Expand the error expectation:

$$\begin{aligned}\mathbb{E}[\|\tilde{g} - g^*\|^2] &= \mathbb{E}[\|g_b + \lambda g_a - g^*\|^2] \\ &= \underbrace{\|\lambda \mathbb{E}[g_a] - \delta_b\|^2}_{\text{Bias}} + \underbrace{d\sigma_b^2 + \lambda^2 d\sigma_a^2}_{\text{Variance}}\end{aligned} \tag{B.1}$$

where we use the assumptions $\mathbb{E}[g_b] = g^* - \delta_b$, $\mathrm{Var}(g_b) = \sigma_b^2 I_d$, and $\mathrm{Var}(g_a) = \sigma_a^2 I_d$. Differentiating with respect to λ gives

$$\nabla_\lambda \mathbb{E}\left[\|\tilde{g} - g^*\|^2\right] = -2\langle \delta_b, \mathbb{E}[g_a]\rangle + 2\lambda\left(\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2\right), \tag{B.2}$$

which leads to the optimal coefficient

$$\lambda^* = \frac{\langle \delta_b, \mathbb{E}[g_a]\rangle}{\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2}. \tag{B.3}$$

We derive a practical approximation for the optimal coefficient λ^* through the following steps.

$$\begin{aligned}\lambda^* &= \frac{\|\delta_b\|\|\mathbb{E}[g_a]\|\cos\theta}{\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2} \\ &= \frac{\|g^* - \mathbb{E}[g_b]\|\|\mathbb{E}[g_a]\|\cos\theta}{\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2} \\ &= \frac{\sqrt{\|g^*\|^2 + \|\mathbb{E}[g_b]\|^2 - 2g^{*T}\mathbb{E}[g_b]}\|\mathbb{E}[g_a]\|\cos\theta}{\|\mathbb{E}[g_a]\|^2 + d\sigma_a^2}\end{aligned} \tag{B.4}$$

Approximation. In practice, we approximate the population means $\mathbb{E}[g_a], \mathbb{E}[g_b]$ by the current mini-batch gradients g_a, g_b, *i.e.*, $\mathbb{E}[g_a] \approx g_a$, $\mathbb{E}[g_b] \approx g_b$. Assuming *Approx* is a satisfactory via STE-based approximation, we can have baseline gradients g_b and better gradients g^* having comparable magnitudes, and a small angle α between them:

$$\|g_b\| \approx \|g^*\|, \cos\alpha \to 1 \tag{B.5}$$

Substituting equation B.5 into equation B.4 and letting α denote the angle between g_b and g^*:

$$\lambda^* \approx \frac{\sqrt{2\|g_b\|^2 - 2\|g_b\|^2 \cos\alpha}\|g_a\| \cos\theta}{\|g_a\|^2}$$
$$= \frac{\sqrt{2(1-\cos\alpha)}\cos\theta\|g_b\|}{\|g_a\|} \tag{B.6}$$

Given $\cos\alpha \to 1$, the compensator's correction inherently limits η to small magnitudes. Empirical observations also suggest $\sqrt{2(1-\cos\alpha)}\cos\theta$ exhibits small magnitude across our experiments:

$$\sqrt{2(1-\cos\alpha)}\cos\theta \approx \eta \tag{B.7}$$

Combining equation B.6 and equation B.7 yields the final practical form:

$$\lambda^* \approx \eta\frac{\|g_b\|}{\|g_a\|} \tag{B.8}$$

□

C SURGE FOR TOY PROBLEM AND VISUALIZATIONS

We implement a simple yet illustrative toy model to optimize the non-convex Beale function. The original model architecture consists of an input layer, a hidden layer with ReLU activation, and an output layer that produces 2D coordinates. In our experiments, we binarize the first and second linear layer.

Convergence and loss curve of toy model. As illustrated in Fig. A, we compare five training methods under identical initialization: FP (Full-precision network), STE, STE+SURGE, Bi-Real, Bi-Real+SURGE. We also provide a focused three-group subset (FP, STE, STE+SURGE) shown in Fig. B for clearer visualization. Specifically, for each binarized layer (based on STE / Bi-Real), SURGE adds a parallel full-precision layer and merges their outputs. We provide trajectory plot, loss curve, distance to optimum, adaptive scaling factor (λ). It can be seen that SURGE achieves better convergence performance, yielding lower loss compared to the control group without SURGE integration. And the scale factor λ also reaches convergence.

Parameter evolution of toy model. As illustrated in Fig. C, we provide a parameter evolution plot, where we track the Frobenius norms of the binary and auxiliary full-precision weights, as well as the learnable scaling factors α_w, α_a. We also provide a focused three-group subset (FP, STE, STE+SURGE) shown in Fig. D for clearer visualization. As shown in the figure of Frobenius norm of weights, compared to the FP model, all binarized variants lie in a narrow band, but variants with SURGE maintain a slightly larger and more stable weight norm after the initial transient. As shown in the figure of learnable scaling factors, SURGE consistently pushes these scales slightly higher.

Weight distribution. As illustrated in Fig. E, we provide the weight distribution of one layer in ResNet-18 trained with CIFAR-10. It can be seen that weights around zero is less with SURGE than the counterpart without SURGE. The binarization results without SURGE is less robust to any robust disturbance (Xu et al., 2022a), as sign(w) would more frequently flips.

Cosine similarity. As shown in Fig. F, we draw a figure of cosine similarity (on ResNet-20 trained with CIFAR-10) between the gradient of weights of main branch and auxiliary branch, averaged over layers and mini-batches. It can be seen that the cosine similarity is relatively high. During the main training phase, the similarity slowly decreases but remains in the range 0.8-0.9. Towards the very end of training, when the model has already entered a small local basin and gradients become very small, the cosine similarity drops more sharply. This is expected: the full-precision branch can still perform fine-grained adjustments inside the basin, whereas the binary branch is constrained by quantization, so the auxiliary branch likely compensates in different directions.

Noise contrast experiment. As shown in Fig. G, we conducted experiments with added noise on the toy model to more intuitively demonstrate that our compensation is not merely noise. The results show that the convergence process with added noise becomes significantly more volatile, and the final convergence performance is worse.

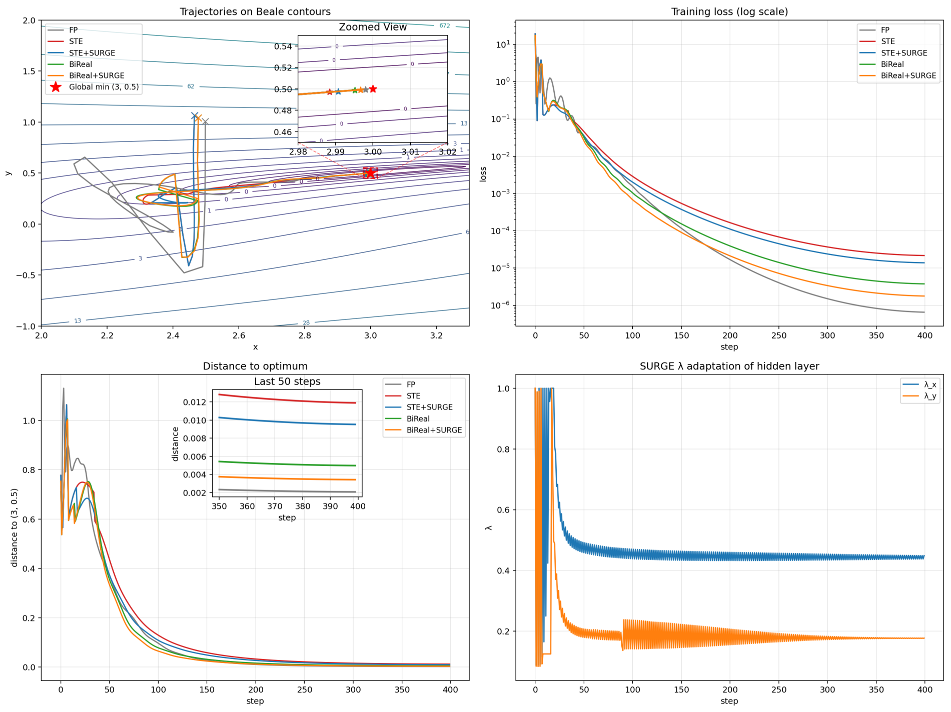

Figure A: Comparison of five methods (FP, STE, STE+SURGE, Bi-Real, Bi-Real+SURGE) on Beale function: (a) trajectories, (b) loss, (c) distance to optimum, (d) SURGE's λ adaptation of hidden layer.

D Dataset, Data Augmentation, and Evaluating Metrics

We evaluate on two standard image classification benchmarks, one object detection benchmark, and one suite of language understanding tasks to demonstrate the effectiveness: CIFAR-10 (Krizhevsky et al., 2009) (10k 32×32 images with random cropping & flipping), ImageNet-1K (Russakovsky et al., 2015) (1.28M training and 50k validation images at 224×224 resolution via center crop), PASCAL VOC (Everingham et al., 2010) (around 16k training and 5k validation images across 20 classes with multi-scale resizing to 1500×900, 1000×600, and 666×400, random flipping at 0.5 ratio), and GLUE (Wang et al., 2018) (covers CoLA, SST-2, MRPC, STS-B, QQP, MNLI (m/mm), QNLI, and RTE, without data augmentation).

We evaluate models using Top-1 and Top-5 accuracy for image classification, mean Average Precision at IoU=0.5 (mAP@0.5) for object detection, and task-specific GLUE metrics following the official protocol: Matthews correlation (MCC) for CoLA; accuracy for SST-2, MNLI (matched/mismatched), QNLI, and RTE; F1/Accuracy for MRPC and QQP; and Spearman correlations for STS-B.

E Implementation Details

E.1 Model Details

CIFAR-10. On CIFAR-10, we evaluate our method with ResNet-18/20 (He et al., 2016) and VGG-Small (Simonyan & Zisserman, 2014). We binarize all convolutional and fully-connected layers except the first and last ones.

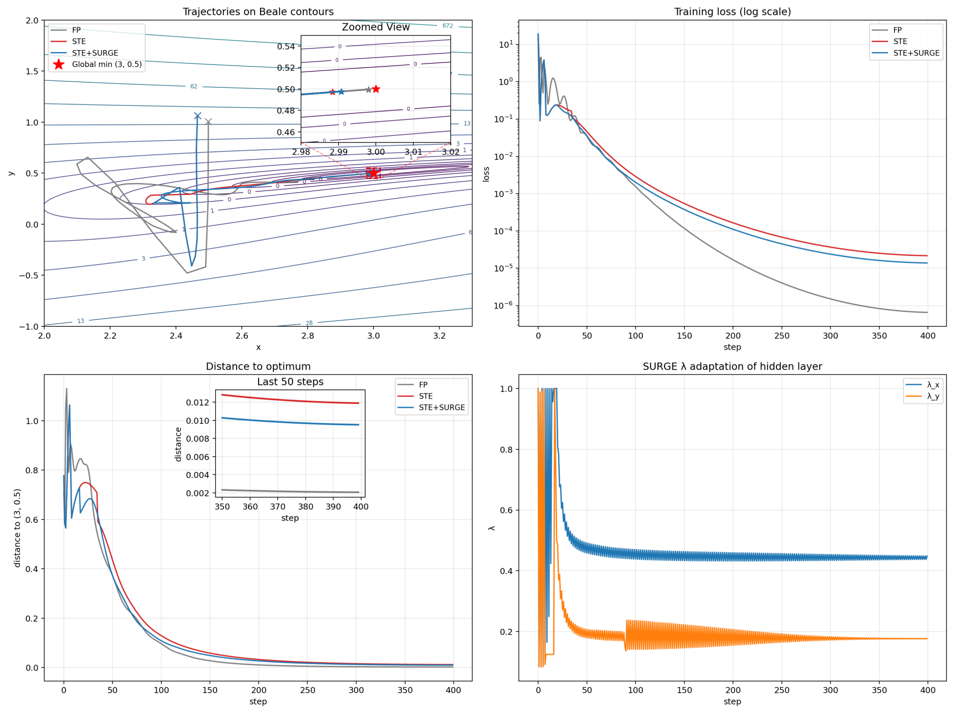

Figure B: Comparison of three methods (FP, STE, STE+SURGE) on Beale function: (a) trajectories, (b) loss, (c) distance to optimum, (d) SURGE's λ adaptation of hidden layer.

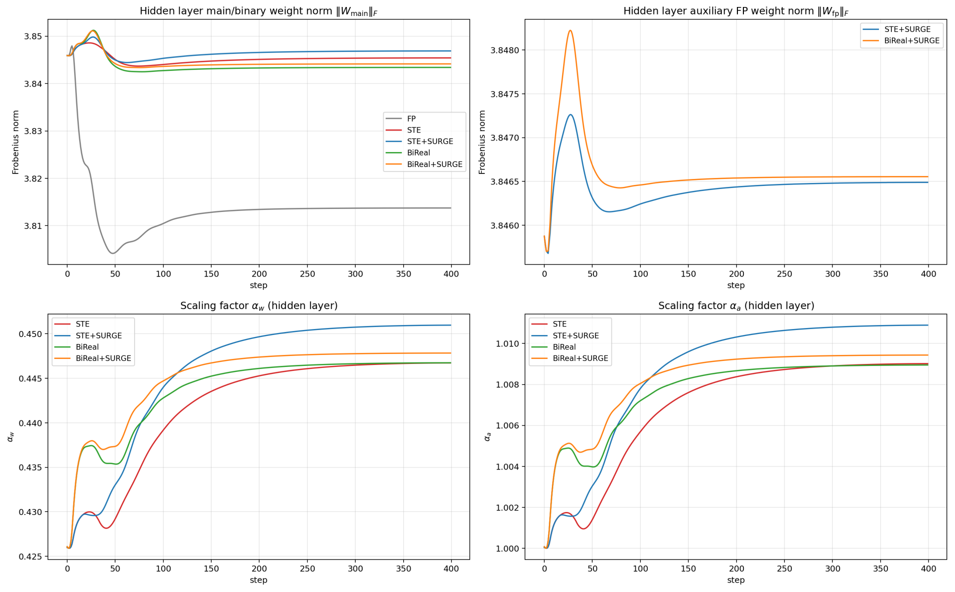

Figure C: Comparison of five methods (FP, STE, STE+SURGE, Bi-Real, Bi-Real+SURGE) on Beale function: (a) weight norm of main branch, (b) weight norm of auxiliary branch, (c) scaling factor of weights, (d) scaling factor of activations.

ImageNet-1K. On ImageNet-1K, we binarize ResNet-18 and retain the first layer, shortcut, and last layer in the networks as real-valued following (Liu et al., 2018b). We adopt the same model modification scheme as described in (Liu et al., 2020).

PASCAL VOC. On PASCAL VOC, we binarize Faster-RCNN with a ResNet-18 backbone. We keep the shortcut, first layer, and the last layer (the 1×1 convolution layer of RPN and an FC layer

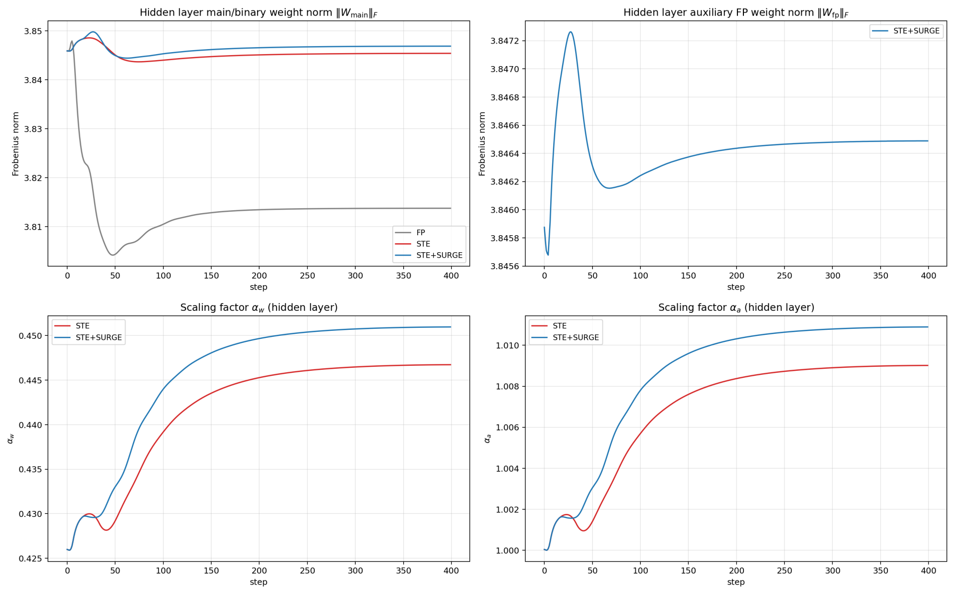

Figure D: Comparison of three methods (FP, STE, STE+SURGE) on Beale function: (a) weight norm of main branch, (b) weight norm of auxiliary branch, (c) scaling factor of weights, (d) scaling factor of activations.

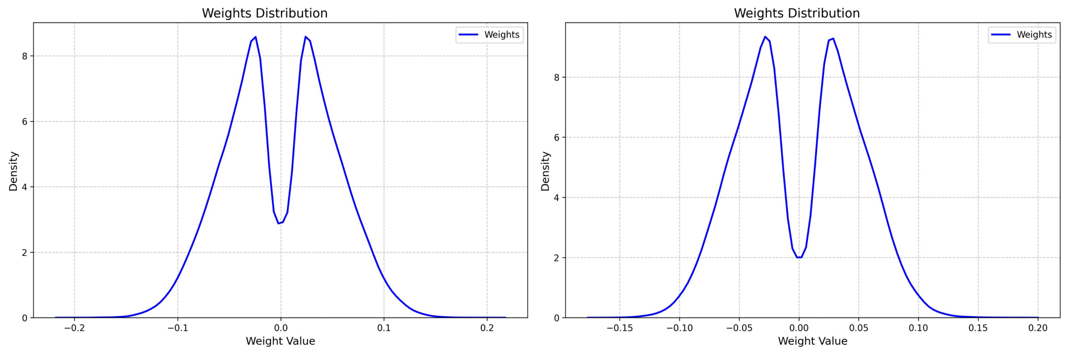

Figure E: Weight distribution comparison of a ResNet-18 layer trained on CIFAR-10. (left) Baseline method; (right) SURGE (ours).

of the bbox head) in the detectors as real-valued after implementing 1-bit CNNs. Following (Wang et al., 2020), we modify the network of ResNet-18 with an extra shortcut and PReLU (He et al., 2015).

GLUE. On GLUE, we evaluate our method with BERT-base (Devlin et al., 2019). We follow the previous work to binarize the word embedding layer, MHA and FFN in transformer layers, but leave full-precision classifier, position embedding layer, and token type embedding layer (Qin et al., 2022; Liu et al., 2022).

E.2 Training Details

CIFAR-10. On CIFAR-10, we train our models from scratch and following the setting in (Xu et al., 2021c), and the base scaling coefficient η is set to 0.01.

ImageNet-1K. On ImageNet-1K, we follow two implementation setups for fair comparison. First, we employ **one-stage training** on ResNet-18 following the setting in (Xu et al., 2022a), using Adam as the optimizer and a weight decay of $1e-5$. The initial learning rate is set to $5e-4$. The model is trained from scratch for 200 epochs with learning rates optimized by the annealing cosine learning rate schedule. Second, we employ **two-stage training** following the setting in (Liu et al., 2020),

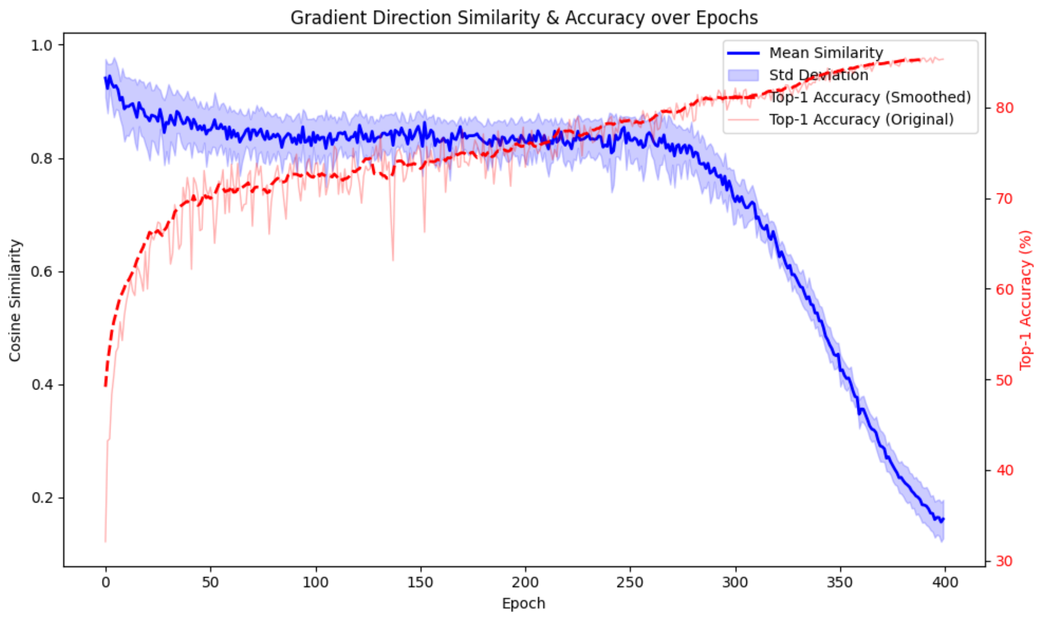

Figure F: Cosine similarity (on ResNet-20 trained with CIFAR-10) between the gradient of weights of main branch and auxiliary branch, averaged over layers and mini-batches

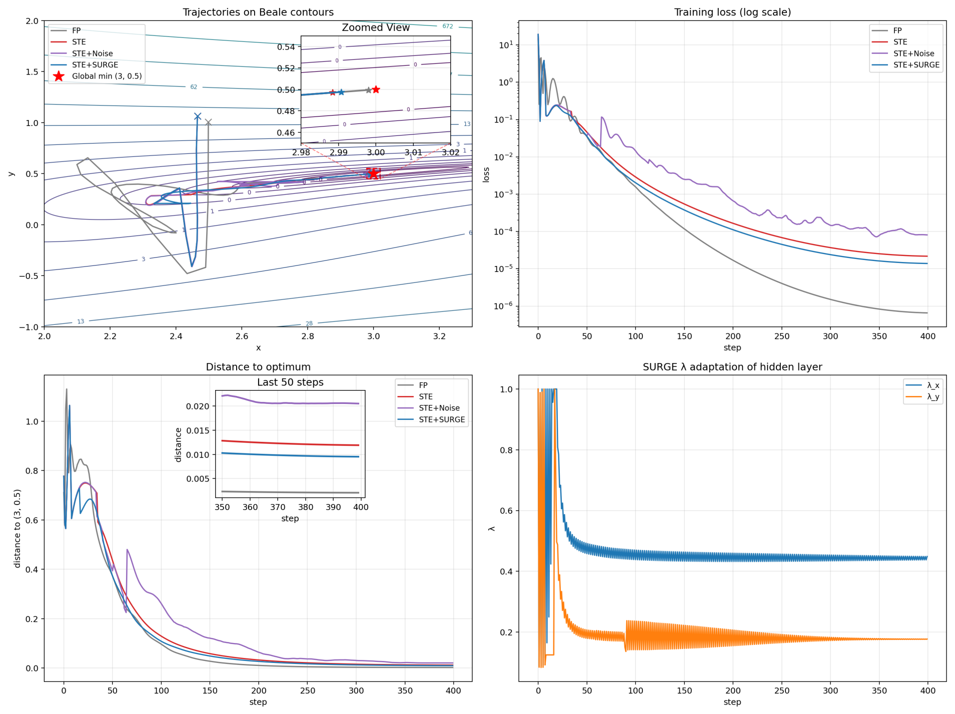

Figure G: Comparison of five methods (FP, STE, STE+Noise, STE+SURGE) on Beale function: (a) trajectories, (b) loss, (c) distance to optimum, (d) SURGE's λ adaptation of hidden layer.

using Adam as the optimizer. The network is supervised by a real-valued ResNet-34 teacher. In the first stage, the model is trained from scratch with binarized activation and real-valued convolution weights. We load the state dict from the first stage, and both the activation and weights are binarized in the second stage. The initial learning rate is set to $5e-4$, the same as one-stage training, and annealed to 0 by a linear descent scheduler. The base scaling coefficient η is set to 0.001.

PASCAL VOC. On PASCAL VOC, we use ImageNet to pre-train the backbone of a 1-bit student, following (Liu et al., 2020). The SGD optimizer is utilized, and the batch size is set as 4 for Faster-RCNN. We train the model in two stages. Only the backbone is binarized at the first stage. Then we binarize all layers in the second stage. Each stage counts 12 epochs. The learning rate is set as 0.004 and decays by multiplying 0.1 in the 9th and 11th epochs following (Xu et al., 2022b). The base scaling coefficient η is set to 0.001.

GLUE. On GLUE, We follow Liu et al. (2022) in adopting the experimental setting of Devlin et al. (2019). We use the Adam as our optimizer, and we take more training epochs for every quantization method on each tasks to have a sufficient training, which is 50 for CoLA, 20 for MRPC, STS-B and RTE, 10 for SST-2 and QNLI, 5 for MNLI and QQP. We distill binary models using full-precision teacher without using multi-distill technique.

For the classification task and language understanding tasks, we performed experiments on 2 NVIDIA RTX 4090 GPUs with 24 GB of memory, along with 216 GB of RAM. For the object detection task, we performed experiments on 4 NVIDIA RTX 2080Ti GPUs with 11 GB of memory, along with 128 GB of RAM.

F Overhead and Deployment Efficiency

SURGE introduces modest additional overhead during training while eliminating any extra inference cost, since the auxiliary branch is discarded after training.

F.1 Training overhead.

CNNs. We conduct a comparison of training time (10 epochs) and memory (batchsize 256/GPU) among SURGE, Bi-Real Net, ReActNet, and RBONN under one-stage training on ImageNet. As shown in Tab. A, our results demonstrate that while the full-precision branch introduces modest overhead (+25% training time, +7.6% memory vs RBONN), SURGE can deliver significant accuracy improvements against other SOTA methods (+0.63% accuracy vs RBONN).

Transformers. We conduct a comparison of training time (1 epoch) and memory consumption between SURGE and baseline (BiT) for BERT quantization on each task of GLUE. Following BiT, we employ task-specific batch sizes during training to optimize performance across different tasks. As shown in Tab. B, SURGE introduces acceptable additional training overhead (+17% avg time, +22% avg memory) while delivering significant accuracy improvements (+1.38% avg accuracy). Notably, when the baseline employs larger batch sizes (32) with substantial memory consumption (12429 MB) on QQP dataset, SURGE adds only minimal additional overhead (+10% memory).

Current BNN deployments predominantly target edge devices where inference efficiency is significant. Consequently, state-of-the-art methods in this domain prioritize two key metrics: (1) achievable accuracy under extreme quantization constraints, and (2) real-world inference latency on resource-limited hardware. Training efficiency remains secondary in established BNN research paradigms. Our method introduces modest additional overhead during training and does not impact deployment.

F.2 Deployment efficiency.

SURGE discards all auxiliary branches after training, maintaining identical resource requirements to standard binary networks while delivering stable accuracy gains. We thank the reviewer for highlighting BitNet's inference method. Based on our prior experience, we implement the 1-bit models on ODROID C4, which has a 2.016 GHz 64-bit quad-core ARM Cortex-A55. By evaluating its real speed in real-world mobile device, the deployment efficiency of SURGE is proven. We leverage the SIMD instruction SSHL on ARM NEON to make the inference library BOLT (Feng, 2021) compatible with SURGE. We compare SURGE to the real-valued backbone in Tab. C. We can see that SURGE's inference speed is substantially faster with the highly efficient BOLT library. For example, the acceleration rate achieves about 4.1× on ResNet18.

Table A: Overhead comparison of training time (10 epochs) and memory (batchsize 256/GPU). * denotes a simple cost-reducing variant.

Method	Training Time (min)	GPU Memory (2 GPUs)	Accuracy (%)
Bi-Real Net	143	19153 MiB×2	38.73
ReActNet	156	20923 MiB×2	45.86
RBONN	160	21005 MiB×2	46.65
SURGE	200	22597 MiB×2	**47.28**
SURGE*	177	22295 MiB×2	47.21

Table B: Comparison of Time, Memory, and Final Accuracy between Baseline and SURGE of training binarized BERT across GLUE tasks

	Method	CoLA	MNLI (m/mm)	MRPC	QNLI	QQP	RTE	SST-2	STS-B
Batch size	———	16	16	8	8	32	8	8	8
Time (1 epoch)/s	Baseline	107	5025	92	2590	3263	61	1583	144
	SURGE	122	6220	109	3050	4110	72	1755	158
Memory (MB)	Baseline	5701	8175	6051	6051	12429	6051	4617	6049
	SURGE	7227	9649	7475	7483	13731	7475	5957	7473
Final accuracy	Baseline	23.56	77.05/77.46	79.41	85.48	85.40	58.12	87.84	67.97
	SURGE	24.11	77.27/77.53	80.64	86.23	87.12	60.65	88.65	71.70

G LIMITATIONS AND FUTURE WORKS

While SURGE improves gradient estimation through its dual-path design, this approach inherently requires the retention of auxiliary full-precision parameters $W_a^{(l)}$ throughout the training phase to enable gradient compensation. This architectural choice introduces two practical considerations: (1) a temporary doubling of parameter memory footprint during backward propagation compared to conventional BNN implementations, and (2) additional computational overhead from parallel path gradient calculations during optimization. In Appendix F, we have conduct experiments to quantify the training overhead of binarizing CNNs and Transformers. Notably, these costs are strictly confined to the training phase. During inference, the auxiliary branches are discarded, restoring the original binary architecture's computational efficiency and memory footprint without any residual overhead.

For architecture, our framework enables exploration of auxiliary structure designs (*e.g.*, via efficient structure design, low-rank decomposition, or saliency-aware layer compensation) to reduce computational overhead. Here we propose a potential method to decrease training overhead. By simply replacing auxiliary convolutions from 3×3 to 1×1 kernels (SURGE* variant in Tab. A), we achieve 14.4% training time reduction (177min vs 200min over 160min), 1.5% memory savings (22295MB vs 22597MB over 21005MB), while still bringing accuracy improvements against other SOTA methods (+0.56% accuracy vs RBONN). This validates that our framework inherently supports efficient re-engineering, and such architectural explorations constitute promising future directions.

H USE OF LARGE LANGUAGE MODELS

We used large language models (LLMs) only to aid and polish writing (*e.g.*, grammar, wording, and minor LaTeX fixes). No algorithms, analyses, results, or claims were generated by LLMs; all technical content and decisions are by the authors. LLMs were not used to create/label data, and

Table C: Deployment efficiency.

Backbone	Method	#bit (W/A)	Size (MB)	Memory Saving	Latency (ms)	Acceleration Rate
ResNet-18	Real-valued	32/32	42.7	–	276.8	–
	SURGE	**1/1**	**1.7**	**25.1×**	**67.8**	**4.1×**

no evaluation items were exposed. Draft snippets were provided as prompts, and all outputs were manually reviewed before inclusion.