A Weight Variation-Aware Training Method for Hardware Neuromorphic Chips

Anonymous authors
Paper under double-blind review

Abstract

Hardware neuromorphic chips that mimic the biological nervous systems have recently attracted significant attention due to their ultra-low power and parallel computation. However, the inherent variability of nano-scale synaptic devices causes a weight perturbation and performance drop of neural networks. This paper proposes a training method to find weight with robustness to intrinsic device variability. A stochastic weight characteristic incurred by device inherent variability is considered during training. We investigate the impact of weight variation on both Spiking Neural Network (SNN) and standard Artificial Neural Network (ANN) with different architectures including fully connected, convolutional neural network (CNN), VGG, and ResNet on MNIST, CIFAR-10, and CIFAR-100. Experimental results show that a *weight variation-aware training method (WVAT)* can dramatically minimize the performance drop on weight variability by exploring a flat loss landscape. When there are weight perturbations, WVAT yields 85.21% accuracy of VGG-5 on CIFAR-10, reducing accuracy degradation by more than 1/10 compared with SGD. Finally, WVAT is easy to implement on various architectures with little computational overhead.

1 Introduction

Deep Neural Networks (DNN) have achieved remarkable breakthroughs in computer vision, automatic driving, and image/voice recognition (LeCun et al., 2015). With this success, neuromorphic technology, which mimics the human nervous system, has recently received significant attention in the semiconductor industry. Compared with the conventional von Neumann architecture which has limitations in power consumption and real-time pattern recognition (Schuman et al., 2017; Indiveri et al., 2015), neuromorphic chips, biologically inspired from the human brain, are new compact semiconductor chips that collocate processing and memory (Chicca et al., 2014; Catherine D. Schuman & Kay, 2022). Therefore, neuromorphic chips can process highly parallel operations and be suitable for real-time recognizing images, videos, and audios with ultra-low power consumption (Indiveri & Liu, 2015).

Neuromorphic chips are also suitable for "Edge AI computing," which process data in edge devices rather than in the cloud at a data center (Nwakanma et al., 2021). In other words, tasks that require a large amount of computation, such as training, are performed in the cloud and inference in edge devices. Traditional cloud AI processing requires sufficient computing power and network connectivity. This means that an enormous amount of data transmission is required, likely increasing data latency and transferring disconnections (Li et al., 2020). It causes severe problems in autonomous driving, robotics, and mobile VR/AR that require real-time processing. Therefore, there is a growing need for data processing on edge devices. Neuromorphic devices are compact, mobile, and energy-efficient, promising candidates for edge computing systems.

However, despite enormous advances in semiconductor integrated circuit (IC) technology, hardware neuromorphic implementation and embedded systems with numerous synaptic devices remain challenging (Prezioso et al., 2015; Esser et al., 2015; Catherine D. Schuman & Kay, 2022). Design considerations such as multi-level state, device variability, programming energy, speed, and array-level connectivity, are required. (Eryilmaz et al., 2015). In particular, nano-electronic device variability is an inevitable issue originating from manufacturing fabrication (Prezioso et al., 2010). Although there are many kinds of nano-electronic devices for neuromorphic systems and in-memory computing–including memristor, flash memory, phase-change memory, and optoelectronic devices–we call them “devices" for readability in this paper.

Device variability causes mapped synaptic weight values in hardware to differ undesirably from software weight. This gap between hardware synapse and software weight makes it challenging to implement neural networks in real-world applications. Many recent studies have reported that device variability can significantly reduce the accuracy of neuromorphic hardware and DNN accelerators (Catherine D. Schuman & Kay, 2022; Peng et al., 2020; Joshi et al., 2020b; Sun & Yu, 2019; Kim et al., 2019; 2018). Although there are various studies to solve this problem, they focus on the unique behaviors of devices (Hennen et al., 2022; vls; Fu et al., 2022). The diversity of devices used to implement neuromorphic hardware results in the customized solutions required for a given device variation. Therefore, the versatility of customized solutions at the device level is limited.

There is a growing need for a hardware-oriented training method to learn parameters robust to device variability. It is widely known that *wide and flat loss landscapes* lead to improved generalization (Keskar et al., 2017; Li et al., 2018). It is natural to think that wide and flat loss landscapes with respect to weight will mitigate the accuracy drop caused by device variability. However, we experimentally confirm that related studies (Izmailov et al., 2018; Wu et al., 2020; Foret et al., 2021) can not significantly reduce the accuracy drop by device variation (Experiments are provided later in section 2). This observation reminds us of the need for a hardware-oriented neural network training method.

Motivated by this, we propose a *weight variation-aware training method (WVAT)* that alleviates performance drops induced by device variability at the algorithmic level rather than the device level. This method explores a wide and flat *weight loss landscape* through the ensemble technique and the hardware-simulated variation-aware update method, which is more tolerant to the software weight perturbation caused by hardware synaptic variability. WVAT can effectively minimize performance drops with respect to weight variations with little additional computational overhead in the training phase. Our contributions include the following:

- For the first time to the best of our knowledge, we investigate and analyze the impact of variations in model parameters on performance in several architectures, including standard Artificial Neural Networks (ANN) and Spiking Neural Networks (SNN), which are suitable for hardware neuromorphic implementation due to event-driven spike properties.
- By exploring the flatter weight loss landscape, we propose WVAT that is tolerant to intrinsic device variability. We introduce an ensemble technique for better generalization and present a intuitive weight update method with a hardware-simulated variation. This method is also efficient for quantization and input noise, which is one of the hardware implementation issues besides weight perturbations.
- We experimentally demonstrate that WVAT achieves nearly similar performance to the typical training method stochastic gradient descent (SGD) while having robustness to variations in model parameters. WVAT is easy to implement with little computational cost.
- By presenting an algorithm-level hardware-oriented training method, we expect that WVAT will help the development of society related to hardware implementation, including neuromorphic chips[1].

2 Background

Many studies have been conducted to develop training methods robust to device variability (Liu et al., 2015; Long et al., 2019; Zhu et al., 2020; Joshi et al., 2020b; Joksas et al., 2022; Huang et al., 2022). Liu et al. (2015) proposed adding a penalty for variations in model parameters to training loss, and Long et al. (2019); Zhu et al. (2020) generated a noise model to reflect device variability during a training phase. Although Long et al. (2019) achieved good performance, this method has a limitation, a binary device (1 bit per cell). However, as mentioned in section 1, the customized solutions for the given device has limitation in applying to the general case.

Recently, there have been many studies investigating the effect of loss landscape on generalization (Garipov et al., 2018; Izmailov et al., 2018; Wu et al., 2020; Foret et al., 2021; Liu et al., 2022). It is

[1]A source code will be available soon.

widely known that a flat loss landscape reduces the generalization gap. Stochastic weight averaging (SWA) (Izmailov et al., 2018) is an ensemble technique that averages the weights as a time axis instead of storing multiple models with different weights. It has been experimentally demonstrated that the ensemble method brings a flatter loss landscape, resulting in a lower test error than SGD. Adversarial weight perturbation (AWP) (Wu et al., 2020) and sharpness-aware minimization (SAM) (Foret et al., 2021) were proposed to improve model generalization by seeking the flat loss landscape. These methods explore the direction of weight perturbation in the worst case and update the weight based on that direction.

AWP AWP proposed a adversarial weight perturbation $\boldsymbol{v}$ based on the generated adversarial examples x_i' for adversarial training:

$$\boldsymbol{v} = \frac{\nabla_{\boldsymbol{v}} \frac{1}{m} \sum_{i=1}^{m} l(f_{\boldsymbol{w}+\boldsymbol{v}}(x_i'), y_i)}{||\nabla_{\boldsymbol{v}} \frac{1}{m} \sum_{i=1}^{m} l(f_{\boldsymbol{w}+\boldsymbol{v}}(x_i'), y_i)||} ||\boldsymbol{w}||$$

where y_i is label, and m is batch size. $f_{\boldsymbol{w}}(\cdot)$ is neural network with weight $\boldsymbol{w}$, and l is standard classification loss.

SAM SAM seeks model parameters whose entire neighborhoods have uniformly low training loss value. Neighborhood $\epsilon(w)$—maximizing loss value— is given by the solution of a classical dual norm problem:

$$\epsilon(\boldsymbol{w}) = \rho \, \mathrm{sign}(\nabla_{\boldsymbol{w}} L_S(\boldsymbol{w})) |\nabla_{\boldsymbol{w}} L_S(\boldsymbol{w})|^{q-1} / (||\nabla_{\boldsymbol{w}} L_S(\boldsymbol{w})||_q^q)^{\frac{1}{p}}$$

where $1/p + 1/q = 1$, ρ is neighborhood size, and $L_S(\cdot)$ is training set loss. In the case of $p = 2$, $\epsilon(\boldsymbol{w})$ is a norm of the gradient scaled by ρ.

Both methods have similarities in finding weight perturbations in the gradient-ascent direction, except the perturbation is scaled by a norm of weight (AWP) or neighborhood size (SAM). Although both methods yielded state-of-the-art performance, they have drawbacks in terms of computational overhead . A update rule in AWP and SAM requires two sequential gradient computations, one for obtaining weight perturbation and another for computing the gradient descent update (Liu et al., 2022). This has twice the computational overhead compared with SGD.

The effect of weight perturbation induced by device variability on model performance can also be considered a kind of generalization problem. We start with a perspective that the flat loss landscape will have robustness against variations in model parameters. A naive approach to solve device variability issues is applying techniques related to generalization. Unfortunately, as shown in Figure 1, these techniques did not lead to a significant improvement in weight variation. As mentioned above, since AWP and SAM are similar algorithms in terms of computing weight perturbations, the experiment was conducted with AWP. When there are variations in the weight, an accuracy drop of 36.58% for SGD, 34.17% for SWA, and 14.89% for AWP occurs compared with a case without variation. The accuracy drop caused by device variability is one of the major issues that make hardware implementation a challenge. This motivates us to explore a new hardware-oriented neural network training method. Therefore, we proposed efficient and intuitive WVAT by considering stochastic weight characteristics incurred by device inherent variability in the training phase and in the weight update to flatten the loss landscape.

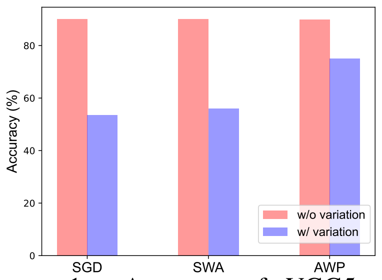

Figure 1: Accuracy of VGG5 on CIFAR-10 using SGD, SWA, and AWP.

3 Weight variation-aware training (WVAT)

Our goal is to find parameters that are tolerant to perturbations of model parameters (e.g., weights) caused by the variability of synaptic hardware devices. We aim to minimize the loss, along with the difference in the loss with respect to perturbed weights. Thus, objective is as follows:

$$\min_{\boldsymbol{w}} [L(\boldsymbol{w}) + (L(\boldsymbol{w} + \boldsymbol{w_v}) - L(\boldsymbol{w}))] \rightarrow \min_{\boldsymbol{w}} L(\boldsymbol{w} + \boldsymbol{w_v})$$

where where $\boldsymbol{w}$ and $\boldsymbol{w_v}$ denotes weight and weight variation, respectively. $L(\cdot)$ represents loss function. In order to explore the flat loss landscapes with respect to weight variation, the difference term $L(\boldsymbol{w} + \boldsymbol{w_v}) - L(\boldsymbol{w})$ should be minimized. When the weight trained in software (cloud) is transferred to a hardware device (edge device), the mapped weight is likely to be different from the software weight due to device variability. This results in performance degradation. For these reasons, here we devise two types of variations that reproduce device variations during a training phase. One is a *hardware-simulated variation (HSV)* reflecting device variability, and the other is a *gradient-ascent variation (GAV)*.

Hardware-simulated variation (HSV)

Assume "analog" synaptic devices, as shown in Figure 2, the layer-wise weight range A is set $[\mu - 3\sigma,\ \mu + 3\sigma]$ across the weight distribution, and we take a weight variation on A.

$$\Delta \boldsymbol{w} = \gamma \mathcal{N}(0,\ A{\sigma_v}^2)$$

where $\Delta \boldsymbol{w}$ is a *hardware-simulated variation (HSV)* to imitate intrinsic hardware device variability, which is a random sample of the same size as $\boldsymbol{w}$ from a Gaussian distribution with mean 0 and variation $A\sigma_v$. γ is a range coefficient, scaling factor, to determine the variation size. In general, when mapping software weights to a synaptic hardware device, clipping method is widely used rather than the full range of software weights ($[w_{min},\ w_{max}]$) due to the memory window limit of the synaptic device (Kwon et al., 2019; Joshi et al., 2020b). Therefore, A is the range of weights that can be expressed in synaptic hardware devices. HSV refers to how much variation occurs within the range that can be expressed in the device.

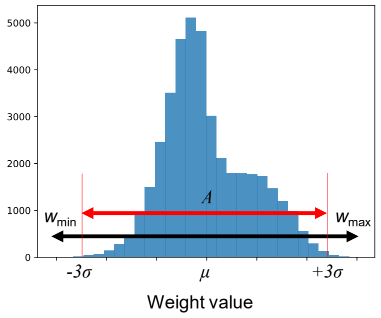

Figure 2: Weight distribution of a layer. A graph indicates the histogram of the software weight. Due to the device variability, the original software weight is perturbed. HSV is drawn from the weight range A to mimic the variation.

Many studies have reported that fabricated synaptic devices have Gaussian distribution (Gong et al., 2018; Boybat et al., 2018; Yu et al., 2013); therefore, we use a Gaussian distribution when generating HSV. In addition, considering that the standard deviation of the fabricated devices is usually $\sim 5\%$ (Joshi et al., 2020a; Wan et al., 2019), we set that a 10% standard deviation was simulated during training for more stable results. For example, σ_v of 10% means that the weight has changed by 10% of A. In order to minimize objective $L(\boldsymbol{w} + \Delta \boldsymbol{w})$ using SGD as a optimizer, differentiating it as follows:

$$\begin{aligned} \nabla_{\boldsymbol{w}} L(\boldsymbol{w} + \Delta \boldsymbol{w}) &= \frac{d(\boldsymbol{w} + \Delta \boldsymbol{w})}{d\boldsymbol{w}} \frac{dL(\boldsymbol{w})}{d\boldsymbol{w}}\bigg|_{\boldsymbol{w}=\boldsymbol{w}+\Delta \boldsymbol{w}} \\ &= 1\nabla_{\boldsymbol{w}} L(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}+\Delta \boldsymbol{w}} \end{aligned}$$

$\nabla_{\boldsymbol{w}} L(\boldsymbol{w} + \Delta \boldsymbol{w})$ can be calculated as the differentiation at the value in which the weight variation occurs by the differentiation of the composite function.

Gradient-ascent variation (GAV) In addition to reproducing HSV during the training phase, the weight variation corresponding to the worst-case—making the greatest the difference term—should also be reflected to find flat loss landscapes. Recalling our objective, this is modified as a maximization problem.

$$\min_{\boldsymbol{w}}[L(\boldsymbol{w}) + \max_{\boldsymbol{v}(\boldsymbol{w})}(L(\boldsymbol{w} + \boldsymbol{w_v}) - L(\boldsymbol{w}))] \rightarrow \min_{\boldsymbol{w}} \max_{\boldsymbol{w_v}} L(\boldsymbol{w} + \boldsymbol{w_v})$$

$L(\boldsymbol{w} + \boldsymbol{w_v})$ can be approximated by first-order Taylor expansion to find the weight variation $\boldsymbol{w_v}$ that maximizes the loss.

$$L(\boldsymbol{w} + \boldsymbol{w_v}) = L(\boldsymbol{w} + \alpha \boldsymbol{v}(\boldsymbol{w})) \approx L(\boldsymbol{w}) + \alpha \boldsymbol{v}(\boldsymbol{w})^T \nabla_{\boldsymbol{w}} L(\boldsymbol{w})$$

where α is a step-length parameter. $\boldsymbol{v}(\boldsymbol{w})^T \nabla_{\boldsymbol{w}} L(\boldsymbol{w})$ is the rate of change in L along the direction $\boldsymbol{v}(\boldsymbol{w})$ at $\boldsymbol{w}$. Therefore, the most rapidly increasing direction is the solution to the problem.

$$\max_{\boldsymbol{v}(\boldsymbol{w})} \boldsymbol{v}(\boldsymbol{w})^T \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \qquad \text{subject to } ||\boldsymbol{v}(\boldsymbol{w})|| = ||\Delta \boldsymbol{w}||$$

$$\boldsymbol{v}(\boldsymbol{w}) = \frac{\nabla L_{\boldsymbol{w}}(\boldsymbol{w})}{||\nabla L_{\boldsymbol{w}}(\boldsymbol{w})||} ||\Delta \boldsymbol{w}||$$

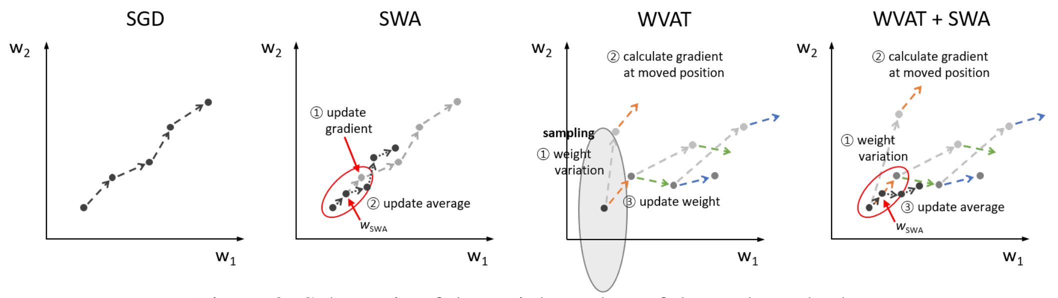

Figure 3: Schematic of the weight update of the each method.

where the magnitude of the weight variation is set to be the same as that of HSV. $\nabla L_{\boldsymbol{w}}(\boldsymbol{w})$ is the steepest ascent direction for a line search method (Nocedal & Wright, 1999). For this reason, we name $\boldsymbol{v}(\boldsymbol{w})$ a *gradient-ascent variation (GAV)*. This is a similar approach to AWP and SAM.

Weight update Weight variation is simulated during training and updates the weight via SGD. The training process is as follows:

$$\text{Weight variation: } \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{w_v}$$
$$\text{Weight update: } \boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla L_{\boldsymbol{w}}(\boldsymbol{w} + \boldsymbol{w_v})$$
$$\text{Weight reverse: } \boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{w_v}$$

where η is a learning rate. In this case, w_v can be one of the two proposed HSV and GAV. Depending on the probability p, it determine whether or not to adding w_v for each batch. If no variance is added, the weight update is equivalent to SGD and no weight reverse is performed.

$$x \sim U(0, 1)$$
$$\text{HSV } \Delta \boldsymbol{w} \text{ is generated if } x < w_{th}$$
$$\text{GAV } \boldsymbol{v}(\boldsymbol{w}) \text{ is generated if } x \geq w_{th}$$

w_{th} is a threshold of what kind of weight variation it will reproduce during training. x is a randomly sampled value from a uniform distribution for each batch. For each batch, x determines what kind of variation will be generated. Figure 3 schematically illustrates the weight update according to each method. SWA averages multiple points along the trajectory of SGD, leading to better generalization than SGD. By applying SWA to WVAT, wider loss landscapes can be explored.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the proposed WVAT on both artificial neural network (ANN) and spiking neural network (SNN) with different architectures, including fully connected (FC), convolutional neural network (CNN), VGG, and ResNet on benchmark datasets (MNIST, CIFAR-10, CIFAR-100). These benchmarks and models have been widely used in hardware implementations (Kim et al., 2018; Long et al., 2019; Zhu et al., 2020; Joshi et al., 2020b; Joksas et al., 2022; Huang et al., 2022; Jung et al., 2022). Edge devices are mainly implemented using small models; moreover, small models are vulnerable to performance degradation due to device variability. Hence, we focus on experiments on small models, including ablation studies and comparisons with SGD, temporal spike sequence learning via backpropagation (TSSL-BP) (Zhang & Li, 2020), and SWA. Although SWA yields better performance via longer training, all method is trained for the same epochs for a fair comparison. We report the mean and standard deviation of test accuracy over 5 runs.

Figure 4 (a) and (b) shows the effect of two hyperparmeters on model performance for VGG5 on CIFAR-10 and VGG16 on CIFAR-100, respectively. We test WVAT for hyperparameter $w_{th} \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. If $w_{th} = 0$, only HSV is reflected, and if $w_{th} = 1$, only GAV is generated during training. For a baseline, we test SGD and SGD Test Time Variation (SGD-TTV), which means when the weight of the trained model perturbs during a test phase. As shown in Figure 4 (a), SGD achieved 90.09% accuracy, while SGD-TTV 53.51% when σ_v of 10% (36.58% performance drop by the variation). In the case of $w_{th} = 1$, only GAV is reproduced

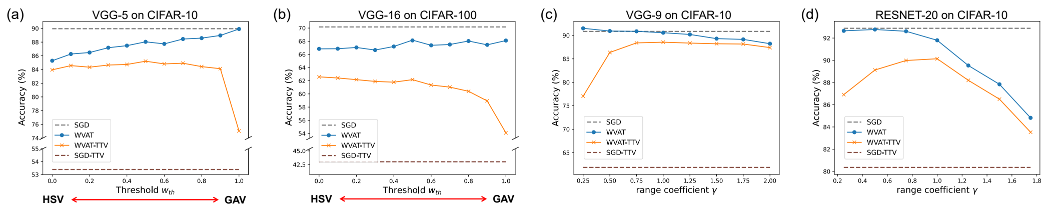

Figure 4: Accuracy of SGD and WVAT. (**a**), (**b**) Effect of HSV and GAV, and (**c**), (**d**) range coefficient γ on model performance.

during the training phase. Although it achieved the highest accuracy, it did not effectively prevent the degradation in accuracy when there was a variation in the weight. On the other hand, in the case of $w_{th} = 0$, WVAT minimized the degradation in accuracy during TTV, but the model performance was poor than SGD. Therefore, we set $w_{th} = 0.5$ as the default value, which achieves nearly the similar performance as SGD and is robust to weight perturbation induced by device variability. Since HSV requires one gradient computation, like SGD, and GAV requires two sequential gradient computations, our proposed WVAT, which is a mixture of HSV and GAV, has less computational overhead compared to AWP and SAM. Through this experiment, we experimentally confirmed that reflecting both HSV and GAV is a key factor in model performance and robustness against stochastic weight characteristics.

Range coefficient γ is a hyperparameter that determines the variation size simulated in training. $\gamma = 1$ means that the variation size during TTV is equally reflected during training. As shown in Figure 4 (c) and (d), HSV with the same variation size as the test minimizes the accuracy drop. We set $\gamma = 1$ as the default value. Engineers can choose w_{th} and γ by considering the trade-offs for accuracy when variations occur and when they do not, according to the characteristics of each device.

4.1 MNIST

For MNIST, we compare the accuracy of the ANN with the FC network and spiking CNN trained by each method in Table 1. FC network are trained using SGD optimizer with learning rate 0.0005. When applying SWA, we use SWA training with a fixed learning rate schedule from scratch. We train spiking-CNNs by TSSL-BP for training SNN, and all models for 100 epochs. For TSSL-BP, we use Adam optimizer with a learning rate of 0.0005 and other settings are the same as in Zhang & Li (2020). Hyperparameters for WVAT are set as default value, $\gamma = 1$, $w_{th} = 0.5$, and $\alpha = 0.1$. For each network architecture, the proposed WVAT achieves nearly similar performance as SGD and SWA and minimizes accuracy degradation when weight variation occurs.

Hardware devices inevitably have inherent variability. Therefore, in most cases, the model performance in software cannot be directly reproduced in hardware implementation. From the hardware implementation point of view, it is preferable to refer to the accuracy when there is weight variation rather than the accuracy when there is no variation. In the case of spiking-CNN1, when $\sigma_v = 10\%$, WVAT yields 96.02% accuracy achieving ≈1/5 reduction of accuracy drop compared with TSSL-BP (16.97% performance drop for TSSL-BP, 3.29% for WVAT).

4.2 CIFAR-10

We test the proposed WVAT on CIFAR-10 in Table 2. For comparison experiments, we use VGG-5, VGG-16, ResNet-20, and ResNet-110 models. VGG-5, VGG-16 and ResNet-20 are trained for 200 epochs using SGD with momentum 0.9, weight decay 0.0005, and an initial learning rate of 0.01. We use Preactivation ResNet-110 in Garipov et al. (2018). ResNet-110 are trained for 150 epochs using SGD with momentum 0.9, weight decay 0.0003, and an initial learning rate of 0.1. When applying SWA, we first run SGD optimizer with a decaying learning rate schedule for 75% of the training budget, and then apply SWA with a fixed learning rate of 0.005 for all models except for 0.01 for ResNet-110. We adopt a hybrid conversion method (Rathi et al., 2020) for SNN implementation. Hyperparameters for WVAT are set as $\gamma = 1$, $w_{th} = 0.5$, and $\alpha = 0.1$ for VGG-5 and 0.01 for VGG-16, ResNet-20, and ResNet-110.

Table 1: Accuracy on MNIST for different model and method. 15C5 means convolution layer with 15 of the 5×5 filters, and P2 pooling layer with 2×2 filters.

Model	Method	Acc (%)	Acc (%) $\sigma_v = 5\%$	Acc (%) $\sigma_v = 10\%$
FC 2-layer[1] (ANN)	SGD	98.13 ± 0.09	97.86 ± 0.14	96.54 ± 0.23
	SWA	**98.56** ± 0.12	98.31 ± 0.71	97.20 ± 1.54
	WVAT	98.47 ± 0.05	**98.39** ± 0.04	**97.98** ± 0.19
Spiking-CNN1[2] (SNN)	TSSL-BP	**99.37** ± 0.02	98.08 ± 0.15	82.40 ± 8.03
	SWA	99.29 ± 0.11	98.62 ± 0.60	90.24 ± 2.34
	WVAT	99.31 ± 0.07	**99.07** ± 0.07	**96.02** ± 0.82
Spiking-CNN2[3] (SNN)	TSSL-BP	99.38 ± 0.05	98.81 ± 0.06	86.17 ± 8.55
	SWA	**99.44** ± 0.03	98.23 ± 0.08	89.73 ± 2.69
	WVAT	99.39 ± 0.03	**99.24** ± 0.08	**95.57** ± 1.87

[1] 784-400-200-10.
[2] 16C5-P2-Concat(32C3, 8C1)-8C1-288.
[3] 15C5-P2-40C5-P2-300.

Table 2: Accuracy on CIFAR-10 for different model and method.

Model	Method	Acc (%)	Acc (%) $\sigma_v = 5\%$	Acc (%) $\sigma_v = 10\%$
VGG-5 (ANN)	SGD	89.96 ± 0.16	79.08 ± 0.95	53.39 ± 3.92
	SWA	**89.97** ± 0.22	80.62 ± 1.51	57.14 ± 2.42
	WVAT	87.76 ± 0.24	**87.05** ± 0.25	**85.21** ± 0.19
VGG-16 (SNN conversion)	SGD	92.68 ± 0.11	90.49 ± 0.08	79.71 ± 1.27
	SWA	**92.77** ± 0.08	91.05 ± 0.16	84.10 ± 0.75
	WVAT	91.79 ± 0.144	**91.20** ± 0.12	**89.25** ± 0.10
ResNet-20 (ANN)	SGD	92.88 ± 0.23	90.57 ± 0.28	80.36 ± 1.17
	SWA	**92.98** ± 0.19	**91.51** ± 0.18	84.54 ± 0.57
	WVAT	91.88 ± 0.11	91.43 ± 0.12	**90.06** ± 0.12
ResNet-110 (ANN)	SGD	95.06 ± 0.17	93.58 ± 0.12	84.68 ± 0.71
	SWA	**95.47** ± 0.06	**94.26** ± 0.09	87.26 ± 0.95
	WVAT	94.45 ± 0.16	93.64 ± 0.13	**90.47** ± 0.12

We experimentally confirm that WVAT minimizes the decrease in accuracy when there is variation induced by device variability in most models. Compared to MNIST, the performance degradation due to weight perturbation is more pronounced. In the case of VGG-5, when $\sigma_v = 10\%$, WVAT yields 85.21% accuracy, while 53.39% for SGD and 57.14% for SWA. WVAT reduces the performance drop by more than 1/10 compared with SGD and SWA. Therefore, we demonstrated that the advantage of WVAT in small models is more significant.

4.3 CIFAR-100

We compare the accuracy of each method on CIFAR-100, and the results are summarized in Table 3. For comparison experiments, we use VGG-16, ResNet-34, and ResNet-110 models. It is basically the same as the experimental setting of CIFAR-10, and the different parts is as follows. Experimental settings for ResNet-34 on CIFAT-100 are the same as ResNet-20 on CIFAR-10. ResNet-110 on CIFAR-100 is trained with an initial learning rate of 0.05. As mentioned in section 4.2, WVAT outperforms other methods not only in the small model but also in Resnet-110. It is important to note that the standard deviation of accuracy using WVAT is also the smallest for each network architecture, which means that the proposed WVAT is tolerant to weight variation.

Table 3: Accuracy on CIFAR-100 for different model and method.

Model	Method	Acc (%)	Acc (%) $\sigma_v = 5\%$	Acc (%) $\sigma_v = 10\%$
VGG-16 (ANN)	SGD	70.17 ± 0.19	63.88 ± 0.41	43.02 ± 0.76
	SWA	$\mathbf{70.46} \pm 0.32$	66.33 ± 0.25	51.87 ± 1.25
	WVAT	68.13 ± 0.12	$\mathbf{66.51} \pm 0.14$	$\mathbf{62.16} \pm 0.11$
ResNet-34 (ANN)	SGD	61.40 ± 0.68	58.73 ± 0.63	49.80 ± 1.04
	SWA	61.28 ± 1.46	59.57 ± 1.38	53.33 ± 1.33
	WVAT	$\mathbf{63.65} \pm 0.43$	$\mathbf{62.49} \pm 0.40$	$\mathbf{59.25} \pm 0.65$
ResNet-110 (ANN)	SGD	76.72 ± 0.55	69.22 ± 0.42	32.68 ± 0.75
	SWA	$\mathbf{78.52} \pm 0.19$	74.33 ± 0.14	52.53 ± 0.32
	WVAT	78.13 ± 0.94	$\mathbf{74.59} \pm 0.87$	$\mathbf{54.25} \pm 0.82$

5 Discussion

5.1 Weight loss landscapes

In this section, an experiment is first conducted to investigate the effect of the proposed WVAT on the geometry of *weight loss landscapes*. We visualize the loss landscapes by plotting the change in loss as the weight moves along the direction of the weight variation $\Delta \boldsymbol{w}$.

$$L(\boldsymbol{w} + \Delta \boldsymbol{w}) = L(\boldsymbol{w} + \gamma \mathcal{N}(0,\, A{\sigma_v}^2))$$

where σ_v varies in the range of $[0, 15]\%$, and moving direction is adjusted with $\gamma = -1$ and $\gamma = 1$. We visualize it as σ_v drawn from 10 different samplings.

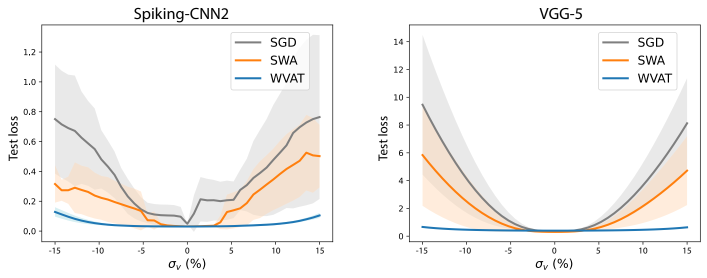

Figure 5: Loss landscapes with respect to model weights. (**left**) Spiking-CNN2 on MNIST, and (**right**) VGG-5 on CIFAR-10.

Figure 5 shows the test loss as a function of the magnitude of the weight variation. Weight loss landscapes become broader and flatter in the order of SGD, SWA, and WVAT, and are uniform for multiple trials. We are taken together with the experimental results in section 4; these experiments show the connection between flat weight loss landscapes and robustness to weight perturbations.

Secondly, we carefully investigate the effect of the proposed WVAT on the weight distribution. As shown in Figure 6 in Appendix A, we verify that WVAT produces larger weights than SGD and SWA. Li et al. (2018) argued that small weights are more sensitive to weight perturbations and make a sharper loss landscape. This claim is consistent with our experimental results. Thus, based on these observations, we demonstrate that WVAT produces larger weights, which makes it less sensitive to weight perturbations and leads to a flatness of the weight loss landscapes.

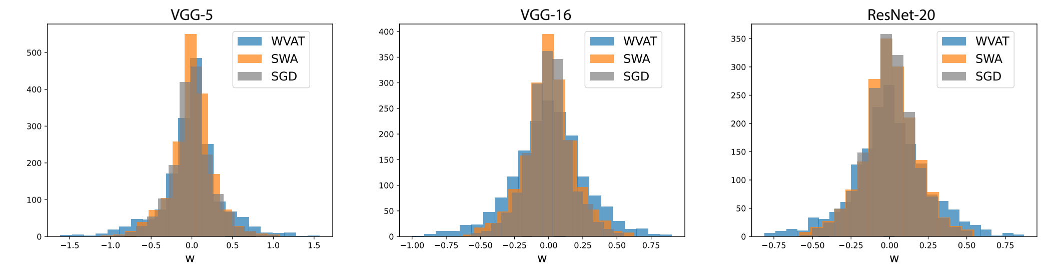

Figure 6: Weight distribution of the first convolutional layer in various network architectures on CIFAR-10.

5.2 Robustness to quantization

Although WVAT is not for a quantization technique, the fact that WVAT is robust to weight perturbation implies that it is also effective against quantization and noise, another hardware implementation issue. Therefore, we evaluate the robustness of WVAT to quantization and input noise. There are various quantization techniques (Zafrir et al., 2019; van Baalen et al., 2022; Jacob et al., 2018; Kirtas et al., 2022) for small models, but post-training quantization is used to simply investigate the effect of quantization.

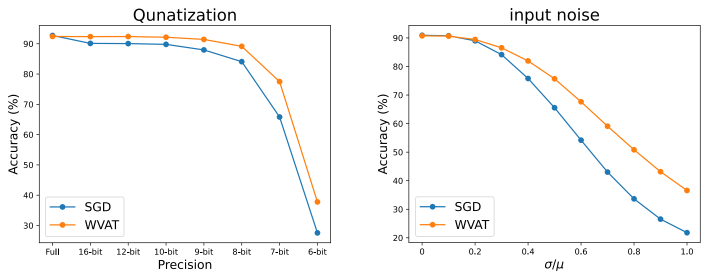

Figure 7: Accuracy of (Left) VGG-16 and (Right) VGG-9 on CIFAR-10.

For quantization and input noise, Figure 7 shows that WVAT provides robustness compared with SGD. We confirm that the proposed method is comprehensively effective for issues for implementing neuro-inspired computing in addition to weight perturbations.

6 Conclusion

This paper proposes a weight variation-aware training method that is robust to weight perturbations incurred by device variability. For the first time to our knowledge, we investigate and analyze the impacts of weight variations on various benchmark datasets and network architectures. The proposed WVAT effectively minimizes performance degradation by more than 1/10 compared to SGD when there is a weight variation. We propose HSV and GAV to mimic weight variations during the training and present a weight update method that considers the stochastic weight characteristics. We experimentally confirm that WVAT is tolerant to weight perturbations by finding the flat loss landscapes with respect to weight. This method is a hardware-oriented training method at the algorithm level rather than a custom solution at the device level to reduce the performance degradation for the stochastic weight characteristic caused by the inherent variability of the device. Therefore, when the weights trained by WVAT are transferred to hardware, accuracy drop due to device variability can be prevented. It is especially effective in small models.

For limitations and future works, the proposed WVAT computes the gradient-ascent variation, requiring two sequential gradient computations, like AWP and SAM, so there is room for computational improvement. Further research is needed on how to achieve similar performance to WVAT with less computational overhead. We think that studies like Liu et al. (2022) can inspire us to solve this problem.

REFERENCES

Irem Boybat, Manuel Le Gallo, SR Nandakumar, Timoleon Moraitis, Thomas Parnell, Tomas Tuma, Bipin Rajendran, Yusuf Leblebici, Abu Sebastian, and Evangelos Eleftheriou. Neuromorphic computing with multi-memristive synapses. *Nature communications*, 9(1):1–12, 2018.

Maryam Parsa J. Parker Mitchell Prasanna Date Catherine D. Schuman, Shruti R. Kulkarni and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2:10–19, 2022. ISSN 2662-8457. doi: https://doi.org/10.1038/s43588-021-00184-y. URL `https://www.sciencedirect.com/science/article/pii/S240595952100059X`.

Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi, and Giacomo Indiveri. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of the IEEE*, 102(9): 1367–1388, 2014. doi: 10.1109/JPROC.2014.2313954.

S. Burc Eryilmaz, Duygu Kuzum, Shimeng Yu, and H.-S. Philip Wong. Device and system level design considerations for analog-non-volatile-memory based neuromorphic architectures. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pp. 4.1.1–4.1.4, 2015. doi: 10.1109/IEDM.2015.7409622.

Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V. Arthur, and Dharmendra S Modha. Backpropagation for energy-efficient neuromorphic computing. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf`.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=6Tm1mposlrM`.

Jingyan Fu, Zhiheng Liao, and Jinhui Wang. Level scaling and pulse regulating to mitigate the impact of the cycle-to-cycle variation in memristor-based edge ai system. *IEEE Transactions on Electron Devices*, 69(4):1752–1762, 2022. doi: 10.1109/TED.2022.3146801.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

Nanbo Gong, T Idé, S Kim, Irem Boybat, Abu Sebastian, V Narayanan, and Takashi Ando. Signal and noise extraction from analog memory elements for neuromorphic computing. *Nature communications*, 9(1):1–8, 2018.

Tyler Hennen, Alexander Elias, Jean-François Nodin, Gabriel Molas, Rainer Waser, Dirk J Wouters, and Daniel Bedau. A high throughput generative vector autoregression model for stochastic synapses. *Frontiers in Neuroscience*, pp. 1423, 2022.

Chenglong Huang, Nuo Xu, Junwei zeng, Wenqing Wang, Yihong Hu, Liang Fang, Desheng Ma, and Yanting Chen. Rescuing reram-based neural computing systems from device variation. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2022.

Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015. doi: 10.1109/JPROC.2015.2444094.

Giacomo Indiveri, Federico Corradi, and Ning Qiao. Neuromorphic architectures for spiking deep neural networks. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pp. 4.2.1–4.2.4, 2015. doi: 10.1109/IEDM.2015.7409623.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Dovydas Joksas, Erwei Wang, Nikolaos Barmpatsalos, Wing H Ng, Anthony J Kenyon, George A Constantinides, and Adnan Mehonic. Nonideality-aware training for accurate and robust low-power memristive neural networks. *Advanced Science*, pp. 2105784, 2022.

Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, Sasidharan Rajalekshmi Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1):1–13, 2020a.

Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, Sasidharan Rajalekshmi Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1):1–13, 2020b.

Seungchul Jung, Hyungwoo Lee, Sungmeen Myung, Hyunsoo Kim, Seung Keun Yoon, Soon-Wan Kwon, Yongmin Ju, Minje Kim, Wooseok Yi, Shinhee Han, et al. A crossbar array of magnetoresistive memory devices for in-memory computing. *Nature*, 601(7892):211–216, 2022.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=H1oyRlYgg`.

Sungho Kim, Meehyun Lim, Yeamin Kim, Hee-Dong Kim, and Sung-Jin Choi. Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network. *Scientific reports*, 8(1):1–7, 2018.

Sungho Kim, Hee-Dong Kim, and Sung-Jin Choi. Impact of synaptic device variations on classification accuracy in a binarized neural network. *Scientific reports*, 9(1):1–7, 2019.

M. Kirtas, A. Oikonomou, N. Passalis, G. Mourgias-Alexandris, M. Moralis-Pegios, N. Pleros, and A. Tefas. Quantization-aware training for low precision photonic neural networks. *Neural Networks*, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2022.09.015. URL `https://www.sciencedirect.com/science/article/pii/S0893608022003598`.

Dongseok Kwon, Suhwan Lim, Jong-Ho Bae, Sung-Tae Lee, Hyeongsu Kim, Chul-Heung Kim, Byung-Gook Park, and Jong-Ho Lee. Adaptive weight quantization method for nonlinear synaptic devices. *IEEE Transactions on Electron Devices*, 66(1):395–401, 2019. doi: 10.1109/TED.2018.2879821.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: https://doi.org/10.1038/nature14539.

En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1): 447–457, 2020. doi: 10.1109/TWC.2019.2946140.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Beiye Liu, Hai Li, Yiran Chen, Xin Li, Qing Wu, and Tingwen Huang. Vortex: Variation-aware training for memristor x-bar. In *Proceedings of the 52nd Annual Design Automation Conference*, pp. 1–6, 2015.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12360–12370, June 2022.

Yun Long, Xueyuan She, and Saibal Mukhopadhyay. Design of reliable dnn accelerator with unreliable reram. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1769–1774. IEEE, 2019.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Cosmas Ifeanyi Nwakanma, Jae-Woo Kim, Jae-Min Lee, and Dong-Seong Kim. Edge ai prospect using the neuroedge computing system: Introducing a novel neuromorphic technology. *ICT Express*, 7(2):152–157, 2021. ISSN 2405-9595. doi: https://doi.org/10.1016/j.icte.2021.05.003. URL `https://www.sciencedirect.com/science/article/pii/S240595952100059X`.

Xiaochen Peng, Shanshi Huang, Hongwu Jiang, Anni Lu, and Shimeng Yu. Dnn+neurosim v2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training, 2020. URL `https://arxiv.org/abs/2003.06471`.

M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, and K. K. Likharev D. B. Strukov. Nanotube devices based crossbar architecture: toward neuromorphic computing. *Nanotechnology*, 21 (17):175202, 2010. doi: https://doi.org/10.1088/0957-4484/21/17/175202.

M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, and K. K. Likharev D. B. Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521:61–64, 2015. doi: https://doi.org/10.1038/nature14441.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *International Conference on Learning Representations*, 2020.

Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware. *CoRR*, abs/1705.06963, 2017. URL `http://arxiv.org/abs/1705.06963`.

Xiaoyu Sun and Shimeng Yu. Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(3):570–579, 2019.

Mart van Baalen, Brian Kahne, Eric Mahurin, Andrey Kuzmin, Andrii Skliar, Markus Nagel, and Tijmen Blankevoort. Simulated quantization, real power savings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2757–2761, June 2022.

Qingzhou Wan, Mohammad T. Sharbati, John R. Erickson, Yanhao Du, and Feng Xiong. Emerging artificial synaptic devices for neuromorphic computing. *Advanced Materials Technologies*, 4(4):1900037, 2019. doi: https://doi.org/10.1002/admt.201900037. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/admt.201900037`.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020.

Shimeng Yu, Bin Gao, Zheng Fang, Hongyu Yu, Jinfeng Kang, and H-S Philip Wong. Stochastic learning in oxide binary synaptic device for neuromorphic computing. *Frontiers in neuroscience*, 7:186, 2013.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pp. 36–39, 2019. doi: 10.1109/EMC2-NIPS53020.2019.00016.

Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33:12022–12033, 2020.

Ying Zhu, Grace Li Zhang, Tianchen Wang, Bing Li, Yiyu Shi, Tsung-Yi Ho, and Ulf Schlichtmann. Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1590–1593. IEEE, 2020.

A Appendix

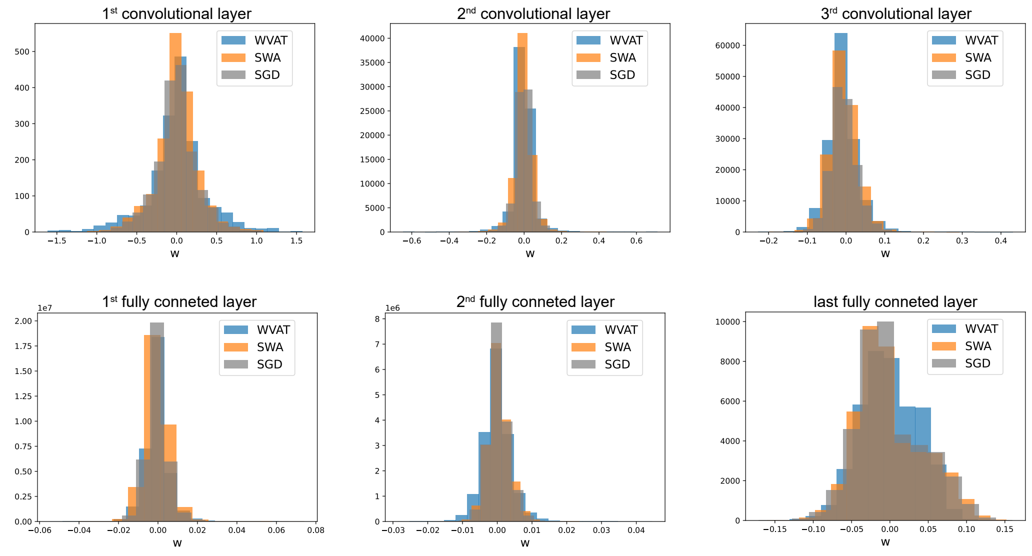

Figure 8: Weight distribution of the entire network layer of VGG-5 on CIFAR-10.

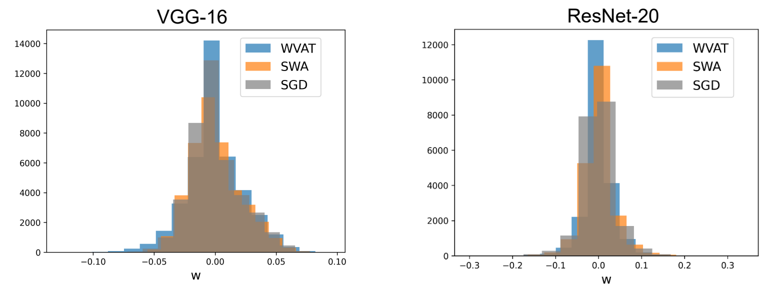

Figure 9: Weight distribution of the last FC layer of VGG-16 and ResNet-20 on CIFAR-10.