Received 3 December 2022, accepted 19 December 2022, date of publication 21 December 2022, date of current version 29 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3231455

RESEARCH ARTICLE

Model Compression via Position-Based Scaled Gradient

JANGHO KIM[1], KIYOON YOO[2], AND NOJUN KWAK[2], (Senior Member, IEEE)
[1]School of Artificial Intelligence, College of Computer Science, Kookmin University, Seoul 02707, Republic of Korea
[2]Department of Intelligence and Information, GSCST, Seoul National University, Seoul 08826, Republic of Korea

Corresponding author: Nojun Kwak (nojunk@snu.ac.kr)

This work was supported in part by the National Research Foundation (NRF) through Korean Government under Grant 2021R1A2C3006659, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) through Korean Government under Grant 2021-0-01343 and Grant 2022-0-00953.

ABSTRACT We propose the position-based scaled gradient (PSG) that scales the gradient depending on the position of a weight vector to make it more compression-friendly. First, we theoretically show that applying PSG to the standard gradient descent (GD), which is called PSGD, is equivalent to the GD in the warped weight space, a space made by warping the original weight space via an appropriately designed invertible function. Second, we empirically show that PSG acting as a regularizer to the weight vectors is favorable for model compression domains such as quantization, pruning, and knowledge distillation. PSG reduces the gap between the weight distributions of a full-precision model and its compressed counterpart. This enables the versatile deployment of a model either as an uncompressed mode or as a compressed mode depending on the availability of resources. The experimental results on CIFAR-10/100 and ImageNet datasets show the effectiveness of the proposed PSG in model compression including an iterative pruning method and the knowledge distillation.

INDEX TERMS Convolutional neural networks, model compression, scaled gradient method, regularization.

I. INTRODUCTION

To reduce the generalization error and induce a prior to the model, many regularization techniques have been proposed [1], [2], [3], [4]. To inject a prior for the specific purpose, we propose a novel regularization method designed for the model compression. This regularizer non-uniformly scales gradients to constrains the weight to a set of compression-friendly grid points. The scale of gradient depends on the position of the weight.

In this work, we propose a new optimizer position-based scaled gradient descent dubbed PSGD. Compared to conventional stochastic gradient descent (SGD), we replace a gradient with position-based scaled gradient. We prove that optimizing the model in the original weight space with PSGD is equal to optimize the model in the warped weight space which is warped by a proposed invertible warping function with a SGD optimizer. This warping function helps to merge the original weights to the desired target positions by scaling the gradients.

PSGD, the scaling gradient elements, is the branch of *variable metric method* [5] which utilizes a positive definite matrix to scale the gradient vector by standing upon the loss function. Unlike, variable metric method, our PSGD only considers the current position of the weight for the scaling gradient elements.

In recent years, deploying a deep neural network (DNN) on restricted edge devices such as smartphones and IoT devices has become a very important issue. For these reasons, reducing bit-width of model weights (quantization) and removing unimportant model weights (pruning), improving the performance of the given model with additional knowledge (knowledge distillation) and proposing for the specific domain have been studied and widely used for applications [10]. We apply the proposed PSG method to the model compression problems such as quantization, pruning and knowledge distillation. Fig 1 shows performances of quantized models using various regularization methods with

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

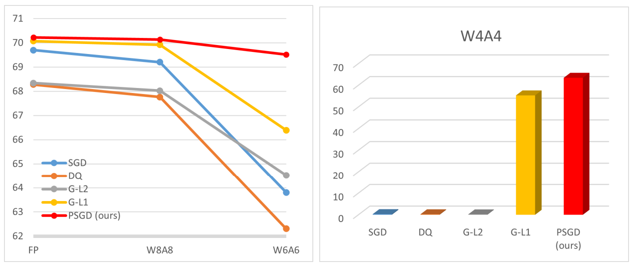


FIGURE 1. Top-1 accuracy on ImageNet with ResNet-18 with various bits. Our PSGD is compared with various methods such as SGD, DQ [6], G-L2 [7] and G-L1 [8]. FP indicates the full-Precision accuracy and W#A# represents the number of bits for weights and activations. More details are in Table 6.

ResNet-18 on ImageNet dataset. PSGD outperforms other regularization methods with large margin.

Since Quantization Aware Training (QAT) methods need a pre-trained model or the entire training dataset for the training, many works have focused on post-training quantization (PTQ) methods that do not require full-scale training [11], [12], [13], [14]. For example, [12] starts with a pre-trained model with only minor modification on the weights by equalizing the scales across channels and correcting biases. Because of inherent discrepancy between the distribution of the pre-trained model and that of quantized model, PTQ methods tried to minimize the distribution gap. Fig. 2 illustrates the fundamental differences between full-precision weights and quantized weights. because of the differences in weight distributions, the quantization error and the classification error increase in accordance with the number of bit-width.

Meanwhile, another line of research in quantization has recently emerged that approaches the task from the initial training phase [8] considered as regularization methods. Compared to PTQ methods, regularization methods tried to reduce the inherent differences by adding the regularizer in the pre-training phase.

Our method is classified as a regularization method. PSGD trains a model from scratch like traditional SGD. Compared to SGD, PSGD focuses to attain a compression friendly model. This model can be effectively pruned or quantized because of its shape of the weight distribution. Consequently, a pre-trained model with PSGD does not require additional post-processing, re-training and accessing the data when the resources are limited. To achieve this goal, PSGD regulates the original weights to merge to a set of grid points by scaling the gradient of weights according to their error between the original weights and the compressed weights (pruned or quantized) (Fig. 3).

This work is the expanded version of our previous research [15]. We additionally verify PSGD with the recent iterative pruning framework. Also, we show that our PSGD as an implicit regularizer not modifying the objective function [16] works well with knowledge distillation which is one of the explicit regularizations. We interpret the geometry of the warped space from PSGD using steepest descent method. Finally, we provide the warped space and the original space distribution analysis.

Our contributions can be summarized as follows:

- We interpret the warped space of PSGD using the steepest descent method with quadratic norm, which tries to make the space wide inversely proportional to quantization error (Eq 26). This phenomenon is also experimentally observed in Sec. V-B3.
- We verify the adaptability of PSGD as an implicit regularizer, which do not modify the objective function by combining iterative pruning methods and a traditional knowledge distillation loss function.
- We provide the analysis of weights in both the warped space and original space in Sec. V-A

II. RELATED WORK

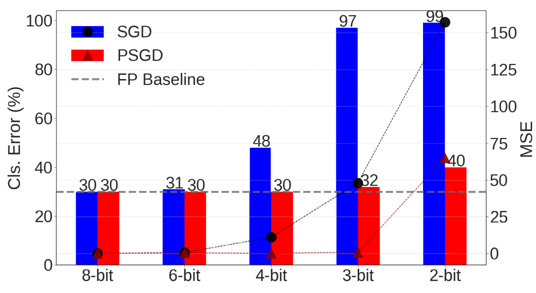
A. QUANTIZATION

Quantization-aware training (QAT) trains the model to attain the quantized model which performs well at the lower bits such as 4,3 and 2 bit. QAT updates the model in the full-precision domain but gradients are calculated in the low-precision domain using the training dataset [17], [18], [19]. To avoid using the whole training dataset and retraining phase, post-training quantization (PTQ) has been researched. These methods do not need a whole training dataset with a simple proposed calculation to consider a resource constraint device [12], [13], [14]. Channel-wise quantization methods require storing quantization parameters per channel and calculations about the quantization bin size [13], [20]. However, layer-wise quantization is more hardware-friendly as they calculate the quantization bin size and store quantization parameters at once per layer [11], [12], [14]. Reference [12] proposes a bias correction and a range equalization of channels, which maintain quantization performance until 8-bit. On the other hand, [14] splits the outliers to reduce the clipping error caused by them. However, these methods still suffer from a significant accuracy drop at the low-bit. References [21] and [22] propose to directly minimize the quantization error using a calibration dataset to achieve higher performance at under 6-bit.

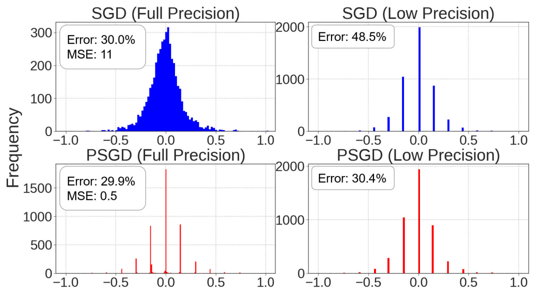
Contrary to previous QAT and PTQ methods, the regularization method has focused on quantization robustness with explicit or implicit regularization terms at the initial training phase. Reference [6] minimizes the Lipshitz constant for robustness against adversarial attacks. Reference [8] proposes an L1 penalty term on the gradients for quantization robustness across different bit widths. This enables a quantization without additional training, dubbed an on-the-fly quantization.

B. PRUNING

Model pruning methods [23], [24], [25], [26], [27], [28], [29], [30] try to prune weights or filters in the model considered



(a) Classification and Quantization Error



(b) Weight Distribution of Full and 4-bit Precision

FIGURE 2. **Results of ResNet-34 on CIFAR-100. (a) Mean-squared quantization error (line) and classification error (bar) across different bits. Blue: SGD, Red: PSGD. (b) Example of weight distribution (Conv2_1 layer [9]) trained with standard SGD and our PSGD. For PSGD, the distribution of the full precision weights closely resembles the low precision distribution, yet maintains its accuracy.**

unimportant units by proposed unique criterion [31], [32]. Many works prune the model in the training phase [33], [34], [35], [36], [37]. Reference [35] proposes a L0 regularization term to train a sparse model. Reference [36] finds a sparse model only using a single shot by the gradient. Similarly, PSGD also does not need a pruning schedule and a retraining phase. PSGD makes a model into a sparse model by using gradient scaling to merge model weights to a zero value.

C. KNOWLEDGE DISTILLATION

Knowledge distillation (KD) is one of the most popular regularization method widely used in model compression domain [3]. This framework uses a larger teacher network's knowledge to boost the performance of a small-size student network. In general, KD encourages the student network to mimic the softened distribution of the teacher network. Another approach named as a feature distillation utilizes the feature map from the teacher network to teach the student network [4], [38].

In doing so, the student network absorbs the knowledge of the teacher by mimicking the logit from the teacher network using the Kullback–Leibler divergence (KL) loss. KD modifies conventional objective terms such as the cross-entropy loss by combining the KL loss. Our PSGD can be combined with this kind of the explicit regularizer because PSGD acts regardless of the objective function. We conduct applying KD with PSGD in Sec. IV-F.

III. PROPOSED METHOD

PSGD regularizes the original weight to converge at the desired target points which can help to perform well in uncompressed and compressed domains. The act of PSGD optimization in the original weight space is equivalent to SGD optimization in the warped weight space. With the invertible function between the original and warped space, PSGD gives the compression-friendly solution by converging a different local minimum compared to the solution of SGD in the original weight space.

A. OPTIMIZATION IN WARPED SPACE

Theorem 1: Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$, $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, be an arbitrary invertible multivariate function that warps the original weight space $\mathcal{X}$ into $\mathcal{Y}$ and consider the loss function $\mathcal{L} : \mathcal{X} \to \mathbb{R}$ and the equivalent loss function $\mathcal{L}' = \mathcal{L} \circ \mathcal{F}^{-1} : \mathcal{Y} \to \mathbb{R}$. Then, the gradient descent (GD) method in the warped space $\mathcal{Y}$ is equivalent to applying a scaled gradient descent in the original space $\mathcal{X}$ such that

$$GD(\boldsymbol{y}, \nabla_{\boldsymbol{y}}^{\mathcal{L}'}) \equiv GD(\boldsymbol{x}, (\mathcal{J}_{\boldsymbol{x}}^{\mathcal{F}})^{-2}\nabla_{\boldsymbol{x}}^{\mathcal{L}}), \tag{1}$$

where $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x})$ and ∇_a^b and $\mathcal{J}_a^b$ respectively denote the gradient and Jacobian of the function b with respect to the variable a.

Proof: Consider the point $\boldsymbol{x}_t \in \mathcal{X}$ at time t and its warped version $\boldsymbol{y}_t \in \mathcal{Y}$. To find the local minimum of $\mathcal{L}'(\boldsymbol{y})$, the standard gradient descent method at time step t in the warped space can be applied as follows:

$$\boldsymbol{y}_{t+1} = \boldsymbol{y}_t - \eta\nabla_{\boldsymbol{y}}^{\mathcal{L}'}(\boldsymbol{y}_t). \tag{2}$$

Here, $\nabla_{\boldsymbol{y}}^{\mathcal{L}'}(\boldsymbol{y}_t) = \frac{\partial \mathcal{L}'}{\partial \boldsymbol{y}}|_{\boldsymbol{y}_t}$ is the gradient and η is the learning rate. Applying the inverse function $\mathcal{F}^{-1}$ to $\boldsymbol{y}_{t+1}$, we obtain the updated point $\boldsymbol{x}_{t+1}$:

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \mathcal{F}^{-1}(\boldsymbol{y}_{t+1}) = \mathcal{F}^{-1}(\boldsymbol{y}_t - \eta\nabla_{\boldsymbol{y}}^{\mathcal{L}'}(\boldsymbol{y}_t)) \\ &= \mathcal{F}^{-1}(\boldsymbol{y}_t) - \eta\mathcal{J}_{\boldsymbol{y}}^{\boldsymbol{x}}(\boldsymbol{y}_t)\nabla_{\boldsymbol{y}}^{\mathcal{L}'}(\boldsymbol{y}_t)\end{aligned} \tag{3}$$

where the last equality is from the first-order Taylor approximation around $\boldsymbol{y}_t$ and $\mathcal{J}_{\boldsymbol{y}}^{\boldsymbol{x}} = \mathcal{J}_{\boldsymbol{y}}^{\mathcal{F}^{-1}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}} \in \mathbb{R}^{n\times n}$ is the Jacobian of $\boldsymbol{x} = \mathcal{F}^{-1}(\boldsymbol{y})$ with respect to $\boldsymbol{y}$. By the chain rule, $\nabla_{\boldsymbol{y}}^{\mathcal{L}'} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = \mathcal{J}_{\boldsymbol{y}}^{\boldsymbol{x}}\nabla_{\boldsymbol{x}}^{\mathcal{L}}$. Because $\mathcal{J}_{\boldsymbol{y}}^{\boldsymbol{x}} = (\mathcal{J}_{\boldsymbol{x}}^{\boldsymbol{y}})^{-1} = (\mathcal{J}_{\boldsymbol{x}}^{\mathcal{F}})^{-1}$, we can rewrite Eq. 3 as

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta(\mathcal{J}_{\boldsymbol{x}}^{\mathcal{F}}(\boldsymbol{x}_t))^{-2}\nabla_{\boldsymbol{x}}^{\mathcal{L}}(\boldsymbol{x}_t). \tag{4}$$

Now Eq. 2 and Eq. 4 are equivalent and Eq. 1 is proved. In other words, the scaled gradient descent (PSGD) in the original space $\mathcal{X}$, whose scaling is determined by the matrix $(\mathcal{J}_{\boldsymbol{x}}^{\mathcal{F}})^{-2}$, is equivalent to gradient descent in the warped space $\mathcal{Y}$. □

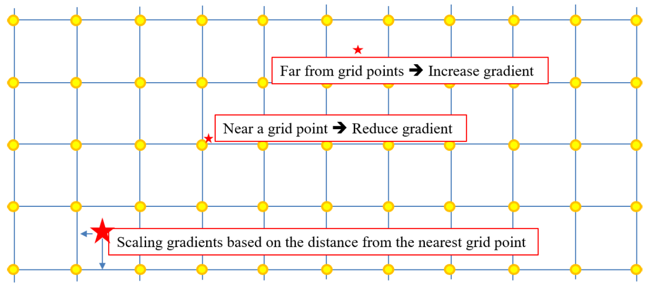

Far from grid points → Increase gradient
Near a grid point → Reduce gradient
Scaling gradients based on the distance from the nearest grid point

FIGURE 3. The main idea of PSGD. Suppose the yellow points indicate the quantization grid in a two-dimensional space. During training in the FP domain, if the position of the weight vector is close to a quantization grid, the gradient of that weight vector is scaled down proportionally to prevent it from escaping. Conversely, if it is distant, the gradient is scaled up so as to accelerate its escape from its original position.This idea is equivalent to multiplying a scaling factor to the gradients based on the distance from the nearest grid point.

B. POSITION-BASED SCALED GRADIENT

In this part, we introduce one example of designing the invertible function $\mathcal{F}(\boldsymbol{x})$ for scaling the gradients. This invertible function should cause the original weight vector $\boldsymbol{x}$ to merge to a set of desired target points $\{\bar{\boldsymbol{x}}\}$. These kinds of desired target weights can act as a prior in the optimization process to constrain the original weights to merge at specific positions. The details of how to set the target points will be deferred to the next subsection.

The gist of weight-dependent gradient scaling is simple. For a given weight vector, if the specific weight element is far from the desired target point, a higher scaling value is applied so as to escape this position faster. On the other hand, if the distance is small, lower scaling value is applied to prevent the weight vector from deviating from the position (See Fig. 3). From now on, we focus on the design of the scaling function for the quantization problem. For pruning, the procedure is analogous and we omit the detail.

1) SCALING FUNCTION

We use the same warping function f for each coordinate $x_i, i \in \{1, \cdots, n\}$ independently, i.e. $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) = [f(x_1), f(x_2), \cdots f(x_n)]^T$. Thus the Jacobian matrix becomes diagonal ($\mathcal{J}_{\boldsymbol{x}}^{\mathcal{F}} = \text{diag}(f'(x_1), \cdots, f'(x_n))$ and our method belongs to the diagonally scaled gradient method.

Consider the following warping function

$$f(x) = 2\,\text{sign}(x - \bar{x})(\sqrt{|x-\bar{x}| + \epsilon} - \sqrt{\epsilon}) + c(\bar{x}) \tag{5}$$

where the target $\bar{x}$ is determined as the closest grid point from x, sign$(x) \in \{\pm 1, 0\}$ is a sign function and $c(\bar{x})$ is a constant dependent on the specific grid point $\bar{x}$ making the function continuous. We introduced $c(\bar{x})$ for making $f(x)$ continuous. If we do not add a constant $c(\bar{x})$, the $f(x)$ has points of discontinuity at every $\{(n+0.5)\Delta | n \in \mathbb{Z}\}$ as depicted in Fig. 4, where Δ represents step size and $n\Delta$ means n-th quantized value identical to $\bar{x}$ corresponding to x. We can calculate the left sided limit and right sided limit at $n\Delta + 0.5\Delta$ using Eq. 5.

$$f(n\Delta + 0.5\Delta^-) = 2(\sqrt{0.5\Delta + \epsilon} - \sqrt{\epsilon}) + c(n\Delta)$$

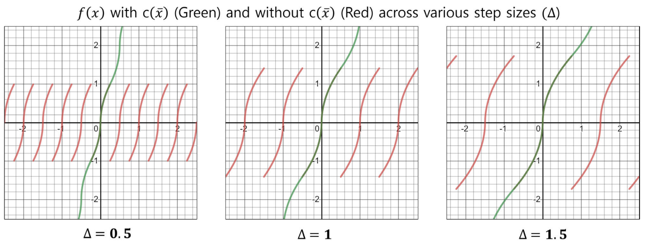


FIGURE 4. Scaling function $f(x)$ for different step size Δ. The red graph depicts $f(x)$ without $c(\bar{x})$ and the green graph depicts $f(x)$ with $c(\bar{x})$ (Eq. 6). Without $c(\bar{x})$, there are points of discontinuity at every $\{(n+0.5)\Delta | n \in \mathbb{Z}\}$. After adding $c(\bar{x})$ to the scaling function $f(x)$, it becomes a continuous function (green).

$$f(n\Delta + 0.5\Delta^+) = -2(\sqrt{0.5\Delta + \epsilon} - \sqrt{\epsilon}) + c((n+1)\Delta)$$

Based on the condition that the left sided limit and the right sided limit should be the same, we can get the following recurrence relation:

$$c((n+1)\Delta) - c(n\Delta) = 4\,(\sqrt{0.5\Delta + \epsilon} - \sqrt{\epsilon}).$$

Using the successive substitution for calculating $c(\bar{x})$, it becomes

$$c(n\Delta) - c(0) = 4n\,(\sqrt{0.5\Delta + \epsilon} - \sqrt{\epsilon}).$$

Setting $c(0) = 0$ and because $n\Delta = \bar{x}$, $c(\bar{x})$ can be calculated as below:

$$c(\bar{x}) = \frac{4\bar{x}}{\Delta}\,(\sqrt{0.5\Delta + \epsilon} - \sqrt{\epsilon}). \tag{6}$$

ϵ is an arbitrarily small constant to avoid infinite gradient. Then, from Eq. 4, the elementwise scaling function becomes $s(x) = \frac{1}{[f'(x)]^2}$ and consequently

$$s(x) = |x - \bar{x}(x)| + \epsilon. \tag{7}$$

Using the elementwise scaling function Eq. 7, the elementwise weight update rule for the PSG descent (PSGD) becomes

$$x_i^{t+1} = x_i^t - \eta s(x_i) \left.\frac{\partial \mathcal{L}}{\partial x_i}\right|_{\boldsymbol{x}^t} \tag{8}$$

where, η is the learning rate.[1] We further elaborate on the geometry of the warped space using the concept of steepest descent in the p-norm in Section III-E.

PSGD operates independent of the type of the loss function as it does not modify the loss term, but rather non-uniformly scales the gradient elements. Therefore, it can be applied to KD loss containing task loss $\mathcal{L}$ (e.g. cross-entropy) and KL loss.Assuming that there are n classes, softmax posterior with temperature $\mathcal{T}$ can be calculated as follows:

$$p_k(\mathbf{z}; \mathcal{T}) = \frac{e^{z_k/\mathcal{T}}}{\sum_j^n e^{z_j/\mathcal{T}}}, \tag{9}$$

[1]We set $\eta = \eta_0 \lambda_s$ where η_0 is the conventional learning rate and λ_s is a hyper-parameter that can be set differently for various scaling functions depending on their range.

where $\mathbf{z}_k$ represents a k-th logit. The temperature value, $\mathcal{T}$, is used to make soft logits for knowledge distillation. We can compute the KL loss between student and teacher network using following equation.

$$KL(z^T||z^S;\mathcal{T}) = \sum_{i=1}^{n} p_k(z^T;\mathcal{T})\log(\frac{p_k(z^T;\mathcal{T})}{p_k(z^S;\mathcal{T})}). \quad (10)$$

where Z^T and Z^S are teacher logit and student logit, respectively. Then, we can use PSGD with KD loss combining task loss and KL loss as below:

$$\mathcal{L}_{KD} = \mathcal{L} + \mathcal{T}^2 \times KL(z^T||z^S;\mathcal{T}) \quad (11)$$

$\mathcal{L}_{KD}$ refers to the KD loss. We multiply $\mathcal{T}^2$ because the decrease rate of the gradient scale is $1/\mathcal{T}^2$. Using KD loss, the update ruls for the PSGD with KD becomes

$${x_i}^{t+1} = {x_i}^t - \eta s(x_i)\left.\frac{\partial \mathcal{L}_{KD}}{\partial x_i}\right|_{\mathbf{x}^t} \quad (12)$$

Applying PSGD at the beginning hinders training the model because of its regularization effect. To relieve this issue, PSGD is applied after a few warm-up epochs. More details are in Sec. IV-A2. The overall process of PSGD is depicted in Algorithm 1.

Algorithm 1 Position-based Scaled Gradient Descent

```
1:  Input: Network weights $\mathbf{x}$, target bit $B$, Warm-up epoch $W$
2:  [ Training phase ]
3:  for Iter $= 1,\ldots,T$ do
4:      if Iter $< W$ then
5:          ${x_i}^{t+1} = {x_i}^t - \eta s(x_i)\left.\frac{\partial \mathcal{L}}{\partial x_i}\right|_{\mathbf{x}^t}$ {Update the model with SGD}
6:      else
7:          ${x_i}^{t+1} = {x_i}^t - \eta s(x_i)\left.\frac{\partial \mathcal{L}}{\partial x_i}\right|_{\mathbf{x}^t}$ {Update the model with PSGD}
8:      end if
9:  end for
10: Output: Quantization friendly model $\mathbf{x}$
11: [ Testing phase ]
12: Quantize network weights $\mathbf{x}$ with target bit-width $B$ and inference with quantized network $\bar{\mathbf{x}}$
```

C. TARGET POINTS

1) QUANTIZATION

In this paper, we use the uniform symmetric quantization method [11] and the per-layer quantization scheme for hardware friendliness. Consider a floating point range [min_x,max_x] of model weights. The weight x is quantized to an integer ranging $[-2^{n-1}+1, 2^{n-1}-1]$ for n-bit precision. Quantization-dequantization for the weights of a network is defined with step-size (Δ) and clipping values. The overall quantization process is as follows:

$$x_Q = Clip(\left\lfloor \frac{x}{\Delta} \right\rceil, -2^{n-1}+1, 2^{n-1}-1),$$

$$\Delta = \frac{max(-min_x, max_x)}{2^{n-1}-1} \quad (13)$$

where $\lfloor\cdot\rceil$ is the round to the closest integer operation and

$$Clip(x,a,b) = \begin{cases} b & \text{if} \quad x > b \\ a & \text{if} \quad x < a \\ x & \text{elsewise.} \end{cases}$$

We can get the quantized weights with the de-quantization process as $\bar{x} = x_Q \times \Delta$ and use this quantized weights for target positions of quantization.

2) PRUNING

For magnitude-based pruning methods, weights near zero are removed. Therefore, we choose zero as the target value (i.e. $\bar{x} = 0$).

D. PSGD FOR DEEP NETWORKS

Many literature focusing on the optimization of DNNs with stochastic gradient descent (SGD) have reported that multiple experiments give consistently similar performance although DNNs have many local minima (e.g. see Sec. 2 of [39]). Reference [40] analyzed the loss surface of DNNs and showed that large networks have many local minima with similar performance on the test set and the lowest critical values of the random loss function are located in a specific band lower-bounded by the global minimum. From this respect, we explain informally how PSGD for deep networks works. As illustrated in Fig. 5, we posit that there exist many local minima (A, B) in the original weight space $\mathcal{X}$ with similar performance, only some of which (A) are close to one of the target points (0) exhibiting high performance also in the compressed domain. As in Fig. 5 left, assume that the region of convergence for B is much wider than that of A, meaning that there exists more chance to output solution B rather than A from random initialization. By the warping function $\mathcal{F}$ specially designed as described above (Eq. 5), the original space $\mathcal{X}$ is warped to $\mathcal{Y}$ such that the areas near target points are expanded while those far from the targets are contracted. If we apply gradient descent in this warped space, the loss function will have a better chance of converging to A'. Correspondingly, PSGD in the original space will more likely output A rather than B, which is favorable for compression. Note that $\mathcal{F}$ transforms the original weight space to the warped space $\mathcal{Y}$ not to the compressed domain.

E. GEOMETRY OF THE WARPED SPACE

In this section, we further illustrate the exact geometry of the warped space when PSGD is applied to quantization. Recall from Eq. 4, Eq. 8, and Eq. 13 that the absolute magnitude of the quantization error is used to scale the gradient elements. This corresponds to left-multiplying a diagonal matrix with the elements determined by the magnitude of the quantization error. We use the concept of p-norm steepest descent [41] to illustrate why this leads to a warped space that induces the weight vectors to merge to the target points. First, we explain some necessary preliminary details for completeness.

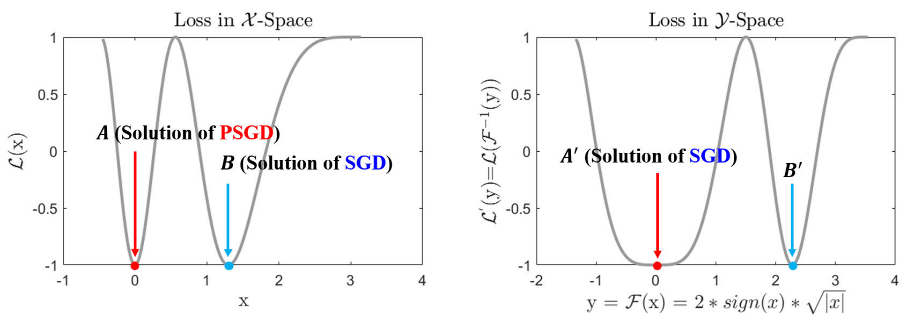


FIGURE 5. Toy example of warping a loss function $\mathcal{L}(x) = \cos((x - 3.07)^2)$. Left denotes the original loss function. Right is drawn by warping the original function by Eq. 5 with the target $\bar{x} = 0$.

1) STEEPEST DESCENT METHOD

For a first-order optimization method, the steepest descent direction, v is determined by minimizing the the first-order Taylor approximation of $L(x + v)$ around x.

$$\mathcal{L}(x+v) \approx \mathcal{L}(x) + \nabla\mathcal{L}(x)^T v \tag{14}$$

Since v can be chosen to have arbitrarily large magnitude in a particular direction, the magnitude is normalized as

$$v_{nsd} = \operatorname{argmin}\{\nabla\mathcal{L}(x)^T v \mid \|v\|_p \leq 1\} \tag{15}$$

Naturally, using different values of p will yield distinct steepest directions. Additionally, other family of norms can also be used such as the quadratic norms, which is defined for a positive-definite matrix $\mathcal{A}$ as

$$\|v\|_{\mathcal{A}} = (v^t \mathcal{A} v^t)^{\frac{1}{2}} = \|\mathcal{A}^{\frac{1}{2}} v\|_2 \tag{16}$$

One can also consider the *unormalized* steepest descent, which scales the normalized steepest descent by the dual norm.

$$v_{sd} = \|\nabla\mathcal{L}(x)\|_* v_{nsd} \tag{17}$$

where $\|\cdot\|_*$ denotes the dual norm

$$\|z\|_* = \sup_x\{|z^t x| \mid \|x\| \leq 1\} \tag{18}$$

For the Euclidean norm ($p = 2$), v_{nsd} corresponds to $-\nabla\mathcal{L}(x)$, which is denoted as gradient descent. Now we present our theorem by interpreting PSGD as steepest descent method in the quadratic norm.

Lemma 1: For a fixed iteration t, the unormalized steepest descent direction in the quadratic norm $\|\cdot\|_{\mathcal{A}}$ is equivalent to the PSG descent direction if the symmetric, positive-definite matrix $\mathcal{A}$ is given by

$$\mathcal{A}(t) = diag(\frac{1}{s(x_i^t)}, \cdots, \frac{1}{s(x_n^t)}) \tag{19}$$

where $s(x)$ is given by Eq. 7 and n is the dimension of the weight vectors.

Proof: First note that the normalized steepest descent in the Euclidean norm is simply given by the negative direction of the gradient scaled by its norm.

$$-\frac{\nabla\mathcal{L}(x)}{\|\nabla\mathcal{L}(x)\|_2} = \operatorname{argmin}\{\nabla\mathcal{L}(x)^T v \mid \|v\|_2 \leq 1\} \tag{20}$$

The steepest descent in the quadratic norm can easily be formulated as above with change of variables.

$$\begin{aligned} v_{nsd} &= \operatorname{argmin}\{\nabla\mathcal{L}(x)^T v \mid \|v\|_{\mathcal{A}} \leq 1\} \\ &= \operatorname{argmin}\{\nabla\mathcal{L}(x)^T v \mid \|\mathcal{A}^{\frac{1}{2}} v\|_2 \leq 1\} \\ &= \operatorname{argmin}\{\nabla\mathcal{L}(x)^T \mathcal{A}^{-\frac{1}{2}} h \mid \|h\|_2 \leq 1\} \end{aligned} \tag{21}$$

where the last equality follows from the change-of-variable $h = \mathcal{A}^{\frac{1}{2}} v$. Then, the descent direction is given by

$$h_{nsd} = -\frac{\mathcal{A}^{-\frac{1}{2}}\nabla\mathcal{L}(x)}{\|\mathcal{A}^{-\frac{1}{2}}\nabla\mathcal{L}(x)\|_2} \tag{22}$$

or equivalently,

$$v_{nsd} = -\frac{\mathcal{A}^{-1}\nabla\mathcal{L}(x)}{\|\mathcal{A}^{-\frac{1}{2}}\nabla\mathcal{L}(x)\|_2} \tag{23}$$

To yield the unormalized descent direction, we compute the dual norm of $\|\nabla\mathcal{L}\|_{\mathcal{A}}$, which is precisely $\sup_x\{|\nabla\mathcal{L}(x)^T x| \mid \|x\|_{\mathcal{A}} \leq 1\} = \|\mathcal{A}^{-\frac{1}{2}}\nabla\mathcal{L}(x)\|_2$. Thus the unormalized descent direction is

$$v_{sd} = -\mathcal{A}^{-1}\nabla\mathcal{L}(x), \tag{24}$$

which written in element-wise for the ith element is equivalent to the PSGD update rule given in Eq. 8.

$$v_{sd}^i = -s(x_i)\frac{\partial\mathcal{L}}{\partial x_i} \tag{25}$$

$\square$

Theorem 2: Given weight spaces $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, and a symmetric, positive-definite matrix $\mathcal{A} \in \mathbb{R}^{n*n}$, let $\mathcal{X}$ and $\mathcal{Y}$ be the weight spaces obtained by PSG descent method and the gradient descent method respectively. Then, the linear transformation from $\mathcal{X}$ to $\mathcal{Y}$ at iteration t is given by

$$\mathcal{T}_t : \mathcal{X} \rightarrow \mathcal{Y} = \mathcal{A}(t)^{\frac{1}{2}} x \tag{26}$$

Thus, for a weight vector x_j^t with small quantization error, the jth basis is expanded inversely proportional to the error, rendering x_j^{t+1} in the vicinity of the target point for a given update.

Proof: For the simplicity of notation, t is omitted below as the proof applies to any fixed t. Consider the loss function defined in $\mathcal{Y}$.

$$\widetilde{\mathcal{L}}(y) = \mathcal{L}(\mathcal{A}^{-\frac{1}{2}}y) = \mathcal{L}(x) \tag{27}$$

The gradient descent direction in y is given by

$$\begin{aligned} v_y &= -\nabla_y \widetilde{\mathcal{L}}(y) = -\frac{\partial \widetilde{\mathcal{L}}(y)}{\partial y} \\ &= -\frac{\partial x}{\partial y}\frac{\partial \mathcal{L}(x)}{\partial x} \\ &= -\mathcal{A}^{-\frac{1}{2}}\nabla_x \mathcal{L} \end{aligned} \tag{28}$$

Applying the inverse transformation of Eq. 26 yields the gradient descent direction in x, which is equivalent to the unormalized steepest descent direction in the qudratic norm given by Eq. 24 in $y \in \mathcal{Y}$.

$$v_x = \mathcal{T}^{-1}(v_y) = -\mathcal{A}^{-1}\nabla_x \mathcal{L} \tag{29}$$

By Lemma 1, this is equivalent to the PSG descent. $\square$

IV. EXPERIMENTS

To verify the effectiveness of PSGD used in model compression, we apply PSGD in three model compression domains including pruning, quantization and knowledge distillation. For pruning, we train the sparse model by setting a target point as 0 without any pruning method and compare this sparse model with L0-regularization [35] and SNIP [36], which are the regularization method and the single-shot pruning method not requiring additional fine-tuning and pruning schedules. Then, we apply our PSGD with iterative pruning methods which require pruning phase and schedule using a magnitude-based pruning criterion.

For quantization, we compare our PSGD with regularization methods which train the model from scratch with regularization [6], [7], [8]. We choose the L1,L2 and Lipschitz regularization methods as the baseline for the original paper of [8]. Then, we also compare with layer-wise PTQ methods that utilize the pre-trained model [12], [14]. Finally, we apply our PSGD with extremely low bits (2,3 bits), various architecture and Adam optimizer [42] for verifying the adaptability.

For knowledge distillation, we conduct experiments to validate the compatibility of PSGD with knowledge distillation by applying it to the KD loss. To show the adaptability of PSGD with other post-training method, we applied the post-training method with PSGD-trained model.

A. IMPLEMENTATION DETAILS

1) HYPER-PARAMETER λ_s

λ_s is the hyper-parameters related to the scaling function (Eq. 8) We tried to find the λ_s which does not bring the performance degradation of uncompressed model, similar to [8]. When we search the hyper-parameter, we utilize two disjoint datasets, the training and validation dataset, from the whole training dataset. After finding the hyper-parameter, we train the model using the whole training dataset and the hyper-parameter. Table 1 and 2 show the values of λ_s used in experiments. The λ_s tended to rise for lower target bit-widths or for higher sparsity ratios. In CIFAR-10, we observe that the same λ_s value yields fair performance across all bit-widths. In our observation, CIFAR-100 and ImageNet dataset need a wide range compared to CIFAR-10.

2) METHODS

All experiments are conducted with the Pytorch framework. For the Single-shot pruning, we used we used ResNet-32 [9] on the CIFAR-100, following the training hyperparameters of [48]. We used released official implementations of [35] and re-implemented [36] for the Pytorch framework. In iterative pruning of Table 4, we followed the same setting of [44].

TABLE 1. λ_s used in the sparse training experiment.

CIFAR-100 & ResNet-32	Sparsity (%)				
	20.0	50.0	70.0	80.0	90.0
λ_s	100	100	200	600	1200

TABLE 2. λ_s used in the quantization experiments.

ResNet-18	ImageNet			CIFAR-10		
	8-bit	6-bit	4-bit	8-bit	6-bit	4-bit
λ_s	500	500	1000	10	10	10

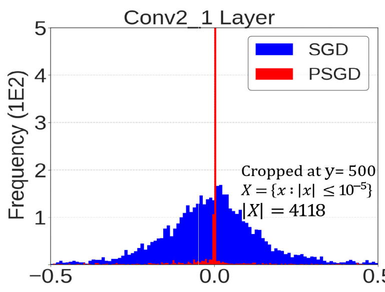


FIGURE 6. The weight distribution of SGD and PSGD models.

TABLE 3. Test accuracy of ResNet-32 across different sparsity ratios (percentage of zeros) on CIFAR-100 after magnitude-based pruning [43] without any fine-tuning.

Method	Sparsity (%)				
	20.0	50.0	70.0	80.0	90.0
SGD	69.43	60.59	15.95	4.70	1.00
L0 Reg. [35]	67.56	64.49	49.73	23.95	2.85
SNIP [36]	**69.68**	68.73	66.76	65.67	60.14
PSGD (Ours)	69.63	**69.25**	**68.62**	**67.27**	**64.33**

TABLE 4. Test accuracy of unstructured pruning for ResNet-20 on CIFAR 10 dataset. All numbers except ours are from [44].

Methods							Sparsity (%)
SNIP [36]	SM [45]	DSR [46]	DPF [44]	DPF + PSGD	AC/DC [47]	AC/DC + PSGD	
88.50±0.13	89.76±0.40	87.88±0.04	90.88±0.07	90.47±0.03	91.13±0.08	**91.20±0.10**	90
84.91±0.25	83.03±0.74	–	88.01±0.30	88.68±0.01	89.76±0.30	**89.93±0.18**	95

TABLE 5. MAC and FLOPS for ResNet20 model on CIFAR10 using a single TITAN RTX GPU.

Model	Pr.ratio	Param. #	MAC	FLOPS
Dense	-	0.275M	41.407M	82.220M
PSGD	90%	0.031M	6.599M	12.605M
	95%	0.018M	4.100M	7.606M

For quantization experiments of Table 6 and 7, we used ResNet-18 and followed [8] settings for CIFAR-10 and ImageNet. For [14], released official implementations were used for experiment. All other numbers are either from the original paper or re-implemented. For fair comparison, all quantization experiments followed the layer-wise uniform symmetric quantization [11] and when quantizing the activation, we clipped the activation range using batch normalization parameters as described in [12], same as [8]. PSGD is applied from the last 15 epochs for ImageNet experiments and from the first learning rate decay epoch for CIFAR experiments. We use additional 30 epochs for PSGD at extremely low bits experiments (Table 8). Also, we tuned the hyper-parameter λ_s for each bit-widths and sparsity. Our search criteria is ensuring that the performance of uncompressed model is not degraded, similar to [8].

3) DATASETS

We use CIFAR-10/100 and the ImageNet datasets for experiments. CIFAR-10 consists of 50,000 training images and 10,000 test images, consisting of 10 classes with 6000 images per class. CIFAR-100 consists of 100 classes with 600 images per class. The ImageNet dataset consists of 1.2 million images. We use 50,000 validation images for the test, which are not included in training samples. We use the conventional data pre-processing steps.[2, 3]

a: ImageNet / CIFAR-10

For ResNet-18, we started training with a L2 weight decay of 10^{-4} and learning rate of 0.1, then decayed the learning rate with a factor of 0.1 at every 30 epochs. Training was terminated at 90 epochs. We only used the last 15 epochs for training the model with PSGD similar to [8]. This means we applied the PSG method after 75 epochs with learning rate 0.001. For extremely low-bits experiments, we did not use any weight decay after 75 epochs. We tuned the hyper-parameters λ_s for target bit-widths. All numbers are results of the last epoch. We used the official code of [14] for comparisons with 0.02 for the Expand Ratio.[4]

b: CIFAR-100

For ResNet-32, the same weight decay and initial learning rate were used as above and the learning rate was decayed at 82 and 123 epoch following [48]. Training was terminated at 150 epoch. For VGG16 with batchnorm normalization (VGG16-bn), we decayed the learning rate at 145 epoch instead. We applied PSG after the first learning rate decay. The first convolutional layer and the last linear layer are quantizedat 8-bit for the 2-bit and the 3-bit experiments. For sparse training, training was terminated at 200 epoch and weight decay was not used at higher sparsity ratio, while all the other training hyperparameters were the same. For [35], we used the official implementation for the results.[5]

B. PRUNING

1) SINGLE-SHOT PRUNING

To figure out that PSGD can train the sparse model with setting the target point at zero, we apply magnitude-based pruning [43] after PSGD training across different sparsity ratios. This setting can be fairly compared with sparsity regularization method and single-shot pruning method because of no need for fine-tuning.

Table 3 shows that our PSGD outperforms two competitive methods in terms of maintaining the performance across different sparsity ratios. Although all methods show promising results at the low sparsity (∼10%), [35] suffers a significant accuracy degradation, where the same phenomenon is observed in another work [49]. Relatively, single-shot pruning [36] maintains the performance at the high sparsity, but PSGD is more potent in making a sparse model. Fig. 6 represents the distribution of weights in SGD- and PSGD-trained models, which explains that the weights are well clustered at the zero target value.

2) ITERATIVE PRUNING

We also consider the iterative pruning case. Comparing single shot pruning methods, iterative pruning gradually increases the sparsity while training. This can help the recovery the

[2]https://github.com/kuangliu/pytorch-cifar
[3]https://github.com/pytorch/examples/blob/master/imagenet/main.py
[4]https://github.com/cornell-zhang/dnn-quant-ocs
[5]https://github.com/AMLab-Amsterdam/L0_regularization

TABLE 6. Test accuracy of regularization methods that do not have post-training process for ResNet-18 on the ImageNet and CIFAR dataset. PSGD@W# indicates the target number of bits for weights in PSGD is #. All numbers except ours are from [8]. At #-bit, PSGD@W# performs the best in most cases.

Method	ImageNet				CIFAR-10			
	FP	W8A8	W6A6	W4A4	FP	W8A4	W6A4	W4A4
SGD	69.70	69.20	63.80	0.30	93.54	85.51	85.35	83.98
DQ Regularization [6]	68.28	67.76	62.31	0.24	92.46	83.31	83.34	82.47
Gradient L2 [7]	68.34	68.02	64.52	0.19	93.31	84.50	84.99	83.82
Gradient L1 [8]	70.07	69.92	66.39	0.22	93.36	88.70	88.45	87.62
Gradient L1 ($\lambda = 0.05$) [8]	64.02	63.76	61.19	55.32	–	–	–	–
PSGD@W8 (Ours)	**70.22**	**70.13**	66.02	0.60	**93.67**	**93.10**	93.03	90.65
PSGD@W6 (Ours)	70.07	69.83	**69.51**	0.29	93.54	92.76	92.88	90.55
PSGD@W4 (Ours)	68.18	67.63	62.73	**63.45**	93.63	93.04	**93.12**	**91.03**

TABLE 7. Comparison with Post-training Quantization methods using ResNet-18 on the ImageNet dataset. Results of DFQ are from [12].

Method	W8A8	W6A6	W4A4
DFQ [12]	69.7	66.3	–
OCS + Best Clip [14]	69.37	66.76	44.3
PSGD (Ours)	**70.13**	**69.51**	**63.45**

TABLE 8. Test accuracy of ResNet-18 on the ImageNet dataset. The first conv layer, the fully-connected layer and activation maps are fixed to 8-bit.

Method	(FP / W3A8)	(FP / W2A8)
SGD	**69.76** / 0.10	**69.76** / 0.10
PSGD (Ours)	66.75 / **66.36**	64.60 / **62.65**

TABLE 9. Test accuracy of ResNet-32 on the CIFAR-100 using Adam optimizer.

Method	FP	W4A32	W4A4
Adam [42]	66.66	55.27	43.5
Adam with PSG	66.80	60.35	51.55

performance via finetuning steps included in iterative pruning training schedule. We bring PSGD to the iterative pruning methods from DPF [44] and AC/DC [47]. Table 4 shows the performance of our method with competitive pruning methods. DPF+PSGD and AC/DC+PSGD refer to combine the DPF with PSGD and AC/DC with PSGD, respectively. PSGD is also very effective in the iterative pruning schemes. PSGD shows promising results by combining iterative pruning method with gradient scaling in the SGD. These results verify that PSGD performs well with only scheduling comparable to the competitive iterative pruning. This can be possible because PSGD only scaled the gradient to regularize the weights to zero which is friendly to pruning. We also report the Multiply and Accumulation (MAC) and FLoating point Operations Per Second (FLOPS) of dense model and PSGD model for comparing the efficiency in Table 5.

TABLE 10. The performances of various architectures with PSGD.

DataSet & Network	Method	(FP / W4A4)	(FP / W3A8)	(FP / W2A8)
CIFAR-100 & VGG16-bn	SGD	73.12 / 63.08	73.12 / 3.44	73.12 / 1.00
	PSGD	73.21 / 70.92	71.85 / 68.28	69.36 / 53.25
DataSet & Network	Method	(FP / W8A8)	(FP / W6A8)	(FP / W4A8)
ImageNet & DeseNet-121	SGD	74.43 / 73.85	74.43 / 70.57	74.43 / 0.36
	PSGD	75.16 / 75.03	75.12 / 74.84	72.60 / 72.26

C. QUANTIZATION

1) REGULARIZATION METHODS

Table 6 provides the results about on-the-fly quantization domain on CIFAR-10 and ImageNet. The setting of on-the-fly quantization evaluates performances across various bit-widths using a single model without any modification. We followed the same setting of [8]. PSGD performs well on CIFAR-10 in every bit-width. In ImageNet dataset, PSGD targeting 8-bit and 6-bit (PSGD@W8 and PSGD@W6) show promising accuracy except 4-bit. Gradient L1 ($\lambda = 0.05$) and PSGD @ W4 maintain the performance of the quantized models even at 4-bit. In general, PSGD outperforms other competitive methods in every bit-width because of quantization friendly weight distributions.

2) POST-TRAINING METHODS

Table 7 shows the performance of Post-training quantization methods and PSGD model. PTQ methods have drastic drops in low bits as depiced in Fig. 1 of the original paper of DFQ [12]. On the other hand, PSGD outperforms OCS at 4-bit about 19% without any post-training method. PSGD also can be combined with PTQ method as stated in Sec. IV-H.

3) EXTREMELY LOW BITS QUANTIZATION

Certainly, the lower bit-width a model is used, the more the performance of quantized model decrease because of its representation power. To verify the expandability of PSGD to extremely low-bits such as 2 and 3-bit, we conducted an experiment targeting 2 and 3-bit except the first, last layers and activation maps (8-bit) in Table 8. This experiment shows that PSGD can be a key solution for the extremely low bit quantization.

TABLE 11. The performance of ResNet-32 on CIFAR-100. Teacher network is ResNet-56. In this experiment, we do not quantize the activation.

Method	(FP / 4-bit)	(FP / 3-bit)	(FP / 2-bit)
PSGD	70.08 / 69.57	68.96 / 68.56	67.16 / 60.76
PSGD + KD	**71.16 / 71.10**	**70.56 / 69.91**	**69.66 / 63.54**

TABLE 12. The performance of ResNet-18 on ImageNet, using ResNet-34 as a teacher network.

Method	(FP / W8A8)	(FP / W6A6)	(FP / W4A4)
PSGD	70.22 / 70.13	70.07 / 69.51	68.18 / 63.45
PSGD + KD	**71.28 / 71.23**	**71.40 / 70.82**	**69.76 / 63.70**

TABLE 13. Comparison with Post-training Quantization methods using ResNet-18 on the ImageNet dataset. PSGD + LAPQ represents LAPQ post-training method with PSGD-pretrained model.

Bits	PSGD	LAPQ [21]	PSGD + LAPQ
W4A4	63.45	60.30	**66.50**

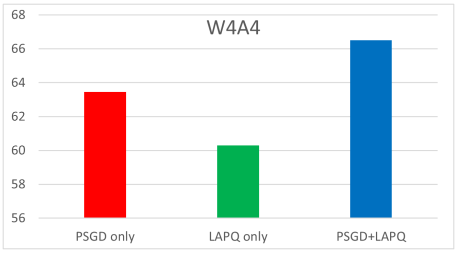


FIGURE 7. Test accuracy of ResNet-18 on ImageNet dataset. PSGD-only (red) and LAPQ-only (green) represent the quantization accuracy of the model trained with PSGD and the post-training accuracy from the pre-trained model with the LAPQ method, respectively. PSGD+LAPQ (blue) means that the quantization accuracy with pre-trained model with PSGD and the LAPQ method.

D. ADAM OPTIMIZER WITH PSG

All previous experiments are conducted with a naive stochastic gradient descent method. To confirm that PSGD can be combined with other optimizers such as Adam, we apply PSGD with Adam optimizer using ResNet-32 on CIFAR-100 dataset. We followed the same setting except for the initial learning rate where 10^{-3} was used. Table 9 shows that PSGD also can be used with another type of optimizer.

E. VARIOUS ARCHITECTURES WITH PSGD

To verify the adaptability of PGSD in the model architecture, in this section, we show the results of applying PSGD to various architectures. Table 10 shows the quantization results of VGG16 [50] with batch normalization on the CIFAR-100 dataset and DenseNet-121 [51] on the ImageNet dataset, respectively.

For DenseNet, we run additional 15 epochs from the pre-trained model to reduce the training time.[6] For fair comparisons in terms of the number of epochs, we also trained for additional 15 epochs for SGD with the same last learning rate (0.001). However, we only observed oscillation in the performance during the additional epochs. Similar to the extremely low-bits experiments, we fixed the activation bit-width to 8-bit.

For VGG16 on the CIFAR-100 dataset, similar tendency in performance was observed with ResNet-32. The 4-bit targeted model was able to maintain its full-precision accuracy, while the model targeting lower bit-widths had some accuracy degradation.

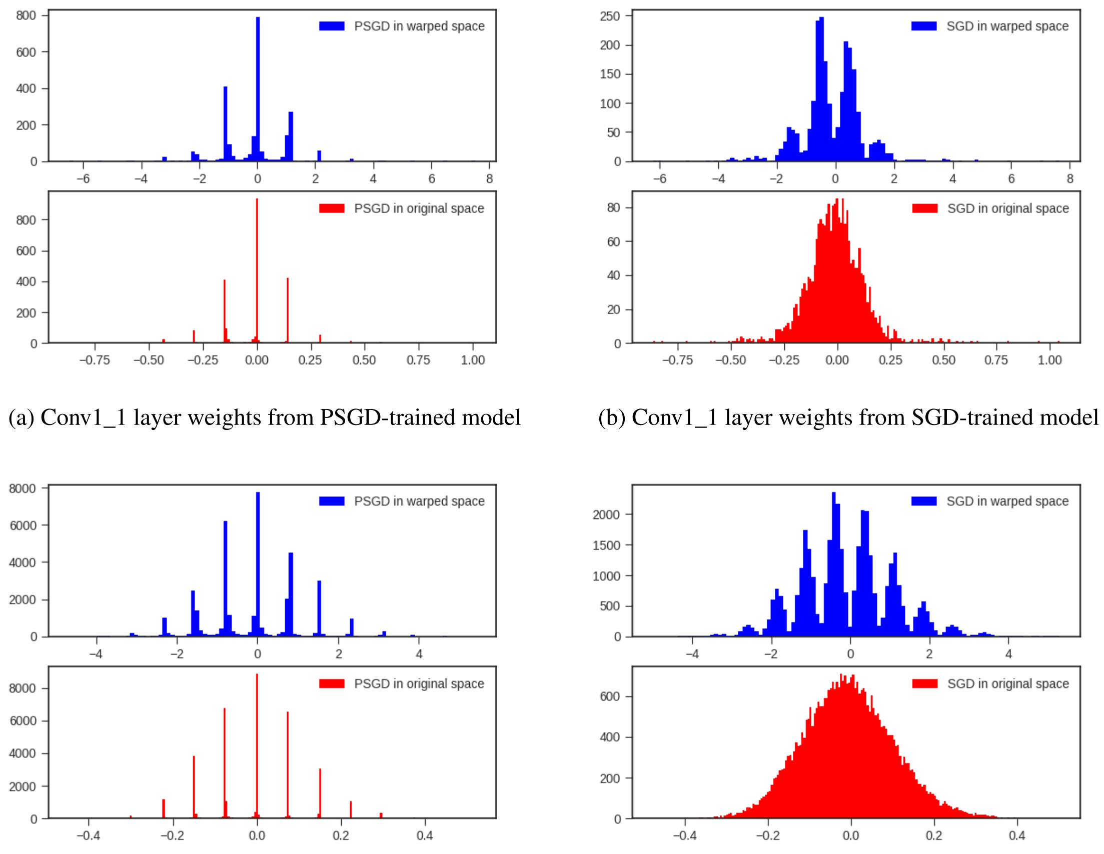
F. KNOWLEDGE DISTILLATION

In this part, we show the adaptability of PSGD, which only manipulates the magnitude of the gradients from the loss function. We apply PSGD with another regularizer, Knowledge Distillation. We follow the update rule (Eq. 12) for quantization, using a KD framework. We utilize a powerful teacher network to train a relatively small student network. We conduct two experiments on CIFAR-100 and ImageNet. In CIFAR-100, we use ResNet-32 as a student and ResNet-54 as a teacher network. In ImageNet, we use ResNet-18 and ResNet-34 as a student and teacher, respectively. Table 11 and 12 show similar tendency. Regardless of bit-width, network, and dataset, Combining KD and PSGD (Eq. 12) outperform using PSGD alone (Eq. 8). From this respect, we validate that PSGD can be used alongside with other regularizer because of its adaptability.

G. QUANTIZATION-AWARE TRAINING VS PSGD

Conventional QAT methods [17], [18], [52] starts with a pre-trained model initially trained with SGD and further update the weights by only considering the low precision weights. In contrast, regularization methods such as our work and [8] starts from scratch and update the full-precision weights *analogous to SGD*. In our work, the sole purpose of PSGD is to find a set of full precision weights that are quantization-friendly so that versatile deployment as low precision (LP) is possible without further operation. Therefore, regularization methods start from the initial training phase analogous to SGD, whereas QAT methods starts with a pre-trained model after the initial training phase such as SGD and PSGD. The purpose of QAT methods is solely focused on LP weights. In general, a coarse gradient is used to update the weights attained by forwarding the LP weights, instead of the FP weights by using the straight-through-estimator (STE) [11]. Additionally, the quantization scheme is modified to include trainable parameters dependent on the low-precision weights and activations. Thus, QAT cannot maintain the performance of full-precision as it only focuses on that of low-precision such as 4 bit-width.

[6]https://download.pytorch.org/models/densenet121-a639ec97.pth



(a) Conv1_1 layer weights from PSGD-trained model (b) Conv1_1 layer weights from SGD-trained model

(c) Conv2_3 layer weights from PSGD-trained model (d) Conv2_3 layer weights from SGD-trained model

FIGURE 8. Weight distributions trained with standard SGD and our PSGD of ResNet-34 on CIFAR-100. Blue: weights in warped space, Red: weights in original space. (a) and (c) are from PSGD-trained model. (b) and (d) are from SGD-trained model. The warped space is calculated by $f(x) = 2\,\mathrm{sign}(x - \bar{x})(\sqrt{|x - \bar{x}| + \epsilon} - \sqrt{\epsilon}) + c(\bar{x})$.

H. POST-TRAINING WITH PSGD-TRAINED MODEL

Our model attains similar full-precision performance with SGD and reasonable performance at low-precision even with naive quantization. Thus, PSGD-trained model can be potentially used as a pre-trained model for QAT or PTQ methods. We performed additional experiments using the model trained with PSGD in Table 7 and Fig. 7 by applying a concurrent PTQ work, LAPQ [21], using the official code.[7] This attains 66.5% accuracy for W4A4, which is more than 3.1% and 6.2% points higher than that of PSGD-only and LAPQ-only respectively as shown in Table 13. This shows that PTQ methods can benefit from using our pretrained model.

V. DISCUSSION

A. THE WARPED WEIGHT SPACE AND THE ORIGINAL WEIGHT SPACE

In this section, we provide the analysis of the weight distribution in both the warped weight space and the original weight space. We already proved that PSGD in the original weight space is equal to SGD in the warped space. And weights in the warped space can be calculated by Eq. 5. Fig. 8 shows weight distributions trained with SGD and PSGD. Based on the Eq. 5, if the quantization error is small, the warped weights are converged at $c(\bar{x})$ which is closely related to $\bar{x}$. The results of Fig. 8 reflect this phenomenon. Warped weights of PSGD are converged at specific points. On the other hand, Warped weights of SGD are spread around specific points compared to PSGD.

[7]https://github.com/ynahshan/nn-quantization-pytorch/tree/master/lapq

B. TOY EXAMPLE

We provide a toy example to give a intuition for the PSGD solution. We trained two models with SGD optimizer and PSGD optimizer under the 2-bit on the MNIST dataset. Each model consists of two layers containing 50 and 20 neurons. We show the weight distribution of the first layer and the eigenvalues of the Hessian matrix.

1) MULTI-MODAL WEIGHT DISTRIBUTION

SGD produces the bell-shaped weight distribution which is not suitable for model compression. On the other hand, PSGD generates multi-moal weight distributions. In this toy example, as we choose a 2-bit for the target bit, three modes

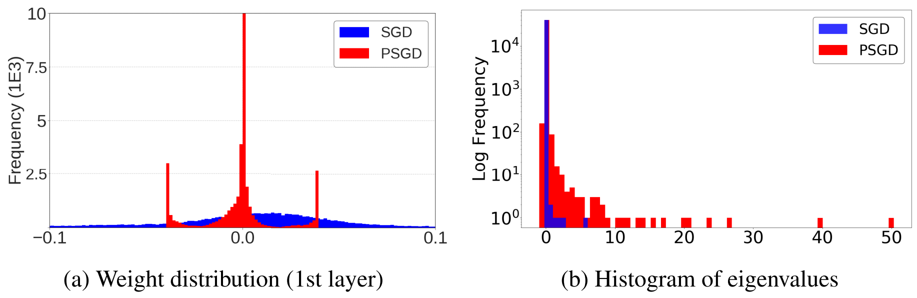


FIGURE 9. Weight distribution and histogram of eigenvalues for MNIST dataset. The two-layered fully connected network consists of 50 and 20 hidden nodes. Target bit of PSGD is 2. Note that both solutions yield relatively small negative eigenvalues ($\lambda > -1$).

exist in the weight distribution of PSGD model as depicted in Fig. 9a. PSGD has nearly the same accuracy with FP ($\sim$96%) at W2A32. However, the accuracy of SGD at W2A32 is about 9%, although the FP accuracy is 97%. This tendency is also shown in Fig. 2b.

2) QUANTIZED AND SPARSE MODEL

In general, SGD produces the bell-shaped weight distribution which is not matched for the quantization. However, PSGD yields a multi-modal distribution. Three target points are used (2-bit) so the weights are merged into three modes as depicted in Fig. 9a. A large proportion of the weights are near zero. Similarly, we note that the sparsity of ResNet-18@W4 shown in Table 6 is 72.4% at LP. This is because symmetric quantization also contains zero as the target point. PSGD has nearly the same accuracy with FP ($\sim$96%) at W2A32. However, the accuracy of SGD at W2A32 is about 9%, although the FP accuracy is 97%. This tendency is also shown in Fig. 2b, which demonstrates that PSGD reduces the quantization error.

3) CURVATURE OF PSGD SOLUTION

In Sec III-D and Fig. 5, we claimed that PSG finds a minimum with sharp valleys that is more compression friendly, but has a less chance to be found. As the curvature in the direction of the Hessian eigenvector is determined by the corresponding eigenvalue [53], we compare the curvature of solutions yielded by SGD and PSGD by assessing the magnitude of the eigenvalues, similar to [54]. SGD provides minima with relatively wide valleys because it has many near-zero eigenvalues and the similar tendency is observed in [54]. However, the weights trained by PSGD have much more large positive eigenvalues, which means the solution lies in a relatively sharp valley compared to SGD. Specifically, the number of large eigenvalues ($\lambda > 10^{-3}$) in PSGD is 9 times more than that of SGD. From this toy example, we confirm that PSG helps to find the minima which are more compression-friendly (Fig 9a) and lie in sharp valleys (Fig. 9b) hard to reach by vanilla SGD. we have also used official code[8] of [55] to qualitatively assess the curvature of Fig. 9, using the same experimental setting, which is depicted in Fig. 10 and it shows a similar tendency. The solution of PSGD is in the more sharp valley than it of SGD.

[8]https://github.com/tomgoldstein/loss-landscape

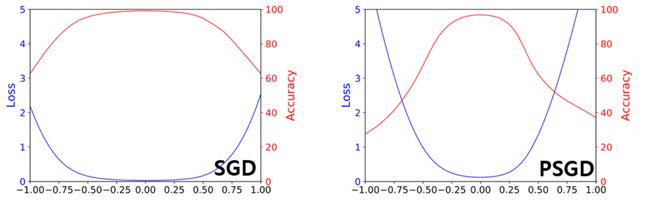


FIGURE 10. Visualizing the loss spaces of Fig. 9 using [55]; Left: Loss space of SGD solution; Right: Loss space of PSGD solution.

VI. CONCLUSION

We propose a new regularization method for model compression. Position-based scaled gradient (PSG) scales the gradient to merge current weights into specific target points such as quantization bins or zero values which are compression-friendly points. Hence, By training the model with PSGD, we can achieve a compression-friendly model. We proved that optimizing the model with position-based scaled gradient descent (PSGD) in the original space is equivalent to optimizing the model in the original space with SGD. The proposed PSGD can be applied at quantization and pruning. Also, we showed that PSGD performs well with knowledge distillation which is an explicit regularization. PSGD will help further research in model compression, including quantization and pruning.

REFERENCES

[1] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B, Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

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

[4] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2760–2769.

[5] W. C. Davidon, "Variable metric method for minimization," *SIAM J. Optim.*, vol. 1, no. 1, pp. 1–17, 1991.

[6] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–14.
[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.
[8] M. Alizadeh, A. Behboodi, M. van Baalen, C. Louizos, T. Blankevoort, and M. Welling, "Gradient ℓ_1 regularization for quantization robustness," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–15.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[10] J. Kim, J. Kim, and N. Kwak, "StackNet: Stacking feature maps for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 242–243.
[11] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.
[12] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1325–1334.
[13] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7948–7956.
[14] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2019, pp. 7543–7552.
[15] J. Kim, K. Yoo, and N. Kwak, "Position-based scaled gradient for model quantization and pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33. Red Hook, NY, USA: Curran Associates, 2020, pp. 20415–20426.
[16] C. Wei, S. Kakade, and T. Ma, "The implicit and explicit regularization effects of dropout," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10181–10192.
[17] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*.
[18] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4852–4861.
[19] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "LSQ+: Improving low-bit quantization through learnable offsets and better initialization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 696–697.
[20] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3009–3018.
[21] Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," 2019, *arXiv:1911.07190*.
[22] M. Nagel, R. Ali Amjad, M. van Baalen, C. Louizos, and T. Blankevoort, "Up or down? Adaptive rounding for post-training quantization," 2020, *arXiv:2004.10568*.
[23] P. Hu, X. Peng, H. Zhu, M. M. S. Aly, and J. Lin, "OPQ: Compressing deep neural networks with one-shot pruning-quantization," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 9, pp. 7780–7788.
[24] J. Kim, J. Yoo, Y. Song, K. Yoo, and N. Kwak, "Dynamic collective intelligence learning: Finding efficient sparse model via refined gradients for pruned weights," 2021, *arXiv:2109.04660*.
[25] J. Kim, S. Chang, and N. Kwak, "PQK: Model compression via pruning, quantization, and knowledge distillation," 2021, *arXiv:2106.14681*.
[26] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, "Only train once: A one-shot neural network training and pruning framework," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 19637–19651.
[27] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.
[28] S. Liu, T. Chen, X. Chen, Z. Atashgahi, L. Yin, H. Kou, L. Shen, M. Pechenizkiy, Z. Wang, and D. C. Mocanu, "Sparse training via boosting pruning plasticity with neuroregeneration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 9908–9922.
[29] E. Tartaglione, A. Bragagnolo, A. Fiandrotti, and M. Grangetto, "Loss-based sensitivity regularization: Towards deep sparse neural networks," *Neural Netw.*, vol. 146, pp. 230–237, Feb. 2022.
[30] M. Lin, R. Ji, S. Li, Y. Wang, Y. Wu, F. Huang, and Q. Ye, "Network pruning using adaptive exemplar filters," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7357–7366, Dec. 2022.
[31] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.
[32] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.
[33] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," 2020, *arXiv:2003.02389*.
[34] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017, *arXiv:1710.01878*.
[35] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through L_0 regularization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
[36] N. Lee, T. Ajanthan, and P. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15.
[37] M. Li, M. Zhao, T. Luo, Y. Yang, and S.-L. Peng, "A compact parallel pruning scheme for deep learning model and its mobile instrument deployment," *Mathematics*, vol. 10, no. 12, p. 2126, Jun. 2022.
[38] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 4619–4625.
[39] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. T. Chayes, L. Sagun, and R. Zecchina, "Entropy-SGD: Biasing gradient descent into wide valleys," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–19.
[40] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. Artif. Intell. Statist.*, 2015, pp. 192–204.
[41] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[43] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[44] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–22.
[45] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," 2019, *arXiv:1907.04840*.
[46] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4646–4655.
[47] A. Peste, E. Iofinova, A. Vladu, and D. Alistarh, "AC/DC: Alternating compressed/decompressed training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 8557–8570.
[48] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 365–382.
[49] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019, *arXiv:1902.09574*.
[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[51] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[52] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, "QKD: Quantization-aware knowledge distillation," 2019, *arXiv:1911.12491*.
[53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[54] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina, "Entropy-SGD: Biasing gradient descent into wide valleys," *J. Stat. Mech., Theory Exp.*, vol. 2019, no. 12, Dec. 2019, Art. no. 124018.

[55] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6389–6399.

JANGHO KIM received the B.S. degree in information and communication engineering from Dongguk University, in 2015, the M.S. degree in computer science and engineering from the Pohang University of Science and Technology (POSTECH), in 2017, and the Ph.D. degree from the Department of Transdisciplinary Studies, Seoul National University, in 2022. He is currently an Assistant Professor with the School of Artificial Intelligence, Kookmin University. His research interests include computer vision, machine learning, and deep learning applied to the image.

KIYOON YOO received the B.S. degree in food science and biotechnology from Seoul National University, in 2019, where he is currently pursuing the Ph.D. degree with the Graduate School of Convergence Science and Technology. His research interests include machine learning, deep learning, and their applications to various areas.

NOJUN KWAK (Senior Member, IEEE) was born in Seoul, South Korea, in 1974. He received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, in 1997, 1999, and 2003, respectively. From 2003 to 2006, he was with Samsung Electronics, Seoul. In 2006, he joined Seoul National University, as a BK21 Assistant Professor. From 2007 to 2013, he was a Faculty Member with the Department of Electrical and Computer Engineering, Ajou University, Suwon, South Korea. Since 2013, he has been with the Graduate School of Convergence Science and Technology, Seoul National University, where he is currently a Professor. His current research interests include feature learning by deep neural networks and their applications in various areas of pattern recognition, computer vision, and image processing.