Front. Comput. Sci., 2025, 19(10): 1910355
https://doi.org/10.1007/s11704-024-40669-3

RESEARCH ARTICLE

Efficient deep neural network training via decreasing precision with layer capacity

Ao SHEN (✉)[1], Zhiquan LAI (✉)[1], Tao SUN[1], Shengwei LI[1], Keshi GE[1,2], Weijie LIU[1], Dongsheng LI (✉)[1]

1 National Key Laboratory of Parallel and Distributed Computing, National University of Defense Technology, Changsha 410073, China

2 College of Computer, National University of Defense Technology, Changsha 410073, China

© The Author(s) 2024. This article is published with open access at link.springer.com and journal.hep.com.cn

Abstract Low-precision training has emerged as a practical approach, saving the cost of time, memory, and energy during deep neural networks (DNNs) training. Typically, the use of lower precision introduces quantization errors that need to be minimized to maintain model performance, often neglecting to consider the potential benefits of reducing training precision. This paper rethinks low-precision training, highlighting the potential benefits of lowering precision: (1) low precision can serve as a form of regularization in DNN training by constraining excessive variance in the model; (2) layer-wise low precision can be seen as an alternative dimension of sparsity, orthogonal to pruning, contributing to improved generalization in DNNs. Based on these analyses, we propose a simple yet powerful technique – DPC (Decreasing Precision with layer Capacity), which directly assigns different bit-widths to model layers, without the need for an exhaustive analysis of the training process or any delicate low-precision criteria. Thorough extensive experiments on five datasets and fourteen models across various applications consistently demonstrate the effectiveness of the proposed DPC technique in saving computational cost (−16.21%–−44.37%) while achieving comparable or even superior accuracy (up to +0.68%, +0.21% on average). Furthermore, we offer feature embedding visualizations and conduct further analysis with experiments to investigate the underlying mechanisms behind DPC's effectiveness, enhancing our understanding of low-precision training. Our source code will be released upon paper acceptance.

Keywords low precision, efficient training, generalization, regularization, bit-width assignment

1 Introduction

The remarkable performance achieved by modern deep neural networks (DNNs) relies on extensive training data and parameters, leading to substantial computational costs in training. Simultaneously, recent advancements in DNNs have spurred an increasing need for intelligent edge devices [1]. Many of these devices operate in dynamic real-world environments, where constrained resources render it impractical to bear the high training costs.

To address these challenges, low-precision training has emerged as a practical approach, employing low precision for both forward and backward propagation during DNN training to optimize the training process, thereby saving on time, memory, and energy costs [2,3]. Typically, the use of low precision leads to significant quantization errors, which should be minimized to maintain model performance. Thus, researchers have explored various solutions to mitigate the impact of quantization errors, such as adapted bit-width assignment [4–12], data format design [13–16], and quantizer optimization [3,17–23].

However, these approaches commonly regard low precision as a source of noise to be minimized or an performance-efficiency trade-off, without considering the potential benefits of low-precision training. Consequently, minimizing the low-precision noise during training requires complex analysis, which introduces additional computational overhead.

Recent findings reveal an opportunity to use low precision for simultaneously improving the efficiency and optimization of DNN training. For example, existing works show that noise can help DNN training theoretically or empirically, motivating us to rethink the role of low precision in DNN training [24,25]. Cyclic Precision Training (CPT) [26] adopts a dynamic precision strategy during training process, which simultaneously improves training efficiency and DNNs' accuracy over state-of-the-art (SOTA) static low-precision training techniques. However, while CPT temporally schedules training precision, it does not investigate methods for assigning varying bit-widths in the spatial dimension of model layers. Meanwhile, random pruning with appropriate layer-wise sparsity ratios may achieve higher test accuracy [27–29]. However, pruning involves the sparsification of parameters, whereas low precision can be understood as the sparsification of data representation.

Received July 3, 2024; accepted October 30, 2024

E-mail: shenao@nudt.edu.cn; zqlai@nudt.edu.cn; dsli@nudt.edu.cn

Motivated by these studies, our work seeks to address the question of how to leverage lower precision spatially to achieve the win-win in both training efficiency and model performance. We systematically investigate the potential benefits of lowering precision. Specifically, low-precision training can theoretically optimize the performance of DNNs and can serve as a regularization technique in DNN training, by constraining excessive variance within the model. We consider the layer-wise low precision as an alternative dimension of sparsity, orthogonal to pruning, which contributes to improved DNNs generalization.

Based on these analyses, we propose Decreasing Precision with layer Capacity (DPC), which determines the precision bound automatically and varies the precision according to layer capacity in a logarithmic schedule patterns. Surprisingly, we find that even without explicitly pursuing the reduction of errors associated with lower precision, low-precision training also has the opportunity to optimize model performance, thereby achieving a win-win for both DNN accuracy and training efficiency.

Thorough experiments on five datasets and fourteen models across various applications (including classification and language modeling) illustrate the consistent effectiveness of the proposed DPC technique in reducing training cost (−16.21%–−44.37%) while achieving comparable or even superior accuracy (up to +0.68%, +0.21% on average). Moreover, we offer feature embedding visualization and further analysis to explore the underlying mechanisms of DPC's effectiveness, contributing to a better comprehension of low-precision training. Overall, our research and findings open up a new design knob for simultaneously improving the optimization and efficiency of DNN training.

Our contribution can be summarized as below:

1. We systematically investigated the potential benefits of low precision in training of DNNs: (i) theoretically demonstrating the enhancement of model generalization due to low precision; (ii) the regularization-like effects imparted by low precision; and (iii) the correlation between low precision and layer capacity.
2. We propose a simple yet powerful technique – DPC, which directly assigns different bit-widths to model layers based on layer capacity, without the need for an exhaustive analysis of the training process or any delicate low-precision criteria.
3. Experimental results validate the surprising effectiveness of DPC, demonstrating improvements in both efficiency (−16.21%–−44.37% training cost) and optimization performance (up to +0.68%, +0.21% on average). The consistency of these results across multiple trials underscores the method's stability and reliability.
4. Further experimental analysis validates the potential benefits brought about by low precision during training, which aligns with our perspective and aids in a deeper understanding of low-precision training.

2 Related work

2.1 Sparsification in DNN

Quantization and pruning are two prominent strategies for sparsification in DNN training, aimed at enhancing the efficiency without significantly compromising accuracy. Quantization involves the use of low-bit precision to represent weights and activations, leading to a substantial reduction in model size and facilitating faster computation at the cost of some accuracy. To address this trade-off, recent studies have explored specialized quantizers and post-training quantization techniques to minimize inaccuracies [30–32]. An emerging direction is mixed-precision quantization, which applies different precision levels to various layers, striking a balance between computational efficiency and model accuracy [33–35]. Pruning, on the other hand, selectively removes redundant or less significant connections within DNNs to simplify the model. It shares similarities with low-precision training in that both aim to optimize DNN performance [27,36]. Additionally, many studies have provided optimization ideas for sparsification in DNN training. Research suggests that the volume of information can be reduced in a way that paradoxically improves model accuracy [37,38]. Understanding the dynamics of DNN training, including the role of noise and the sequential learning of features [24,25,39], provides further insights into how low-precision training might be optimized. These findings collectively invite exploration into new paradigms for more efficient and effective DNN training methodologies.

2.2 Low-precision training

2.2.1 Mitigating low-precision noise in training

Low-precision training, which employs low-bit quantization in both the forward and backward propagation phases, is aimed at enhancing the efficiency of the DNN training. However, the introduction of low precision often brings with it low-precision noise, which can affect the effectiveness of DNN training. Therefore, a key issue is how to reduce the impact of low-precision noise. Many studies [40–46] have analyzed the impact of low precision on DNN training and have proposed techniques to mitigate its negative effects, such as bit-width assignment [4–9,11,12,47], data format design [10,13–16], or quantizer optimization [3,17–23,48,49]. Moreover, to reduce excessive memory consumption during training, various compression techniques [50–52] have been employed to significantly reduce the size of activations. However, the attempts to minimize low-precision noise for different tasks and models often require complex analysis and computation, introducing additional computational overhead and lacking universality, which makes rapid task transfer difficult. At the same time, these methods focus primarily on minimizing errors caused by low precision to better trade-off training efficiency and performance. The potential benefits of low precision are overlooked.

2.2.2 Low precision for optimized DNN training

In contrast to minimizing the low-precision noise in DNN training, several studies have recently argued that low precision can actually enhance DNN training. Yang et al. [2]

have treated precision as a hyperparameter in training. Zhuang et al. [53] have developed a method to train DNNs using both low and high precision to learn from each other. However, these methods require additional computational costs, which can impact the efficiency of these methods. To our knowledge, the most relevant work is CPT [26], which treats precision as analogous to the learning rate in DNN training and has demonstrated that employing variable bit widths during training can aid in locating flatter minima, thereby enhancing the model's generalization. However, CPT focuses solely on temporal precision optimization and does not account for spatial precision considerations.

3 Methodology

In this section, we first introduce our rethinking of low-precision training that motivates us to develop DPC by analogy to regularization techniques and pruning in Section 3.1, and then present the DPC concept in Section 3.2 followed by the precision bounds exploration in Section 3.3, which aims to automate the precision schedule for DPC.

3.1 Rethinking of low-precision training

As mentioned in Section 2.2.1, the motivation for current research focuses on mitigating the impact of low precision in each training iteration. Paradoxically, some studies have achieved superior results with lower precision, despite aiming to align low-precision training with its high-precision counterpart [4,14,41]. In extreme cases, optimized results can even be attained through arbitrary use of low precision. Although some works mention the benefits of low precision in explaining their improvement, systematic research on the benefits is still lacking. The potential benefits of low precision may be underestimated. Considering that previous efforts to minimize the impact of low-precision noise often required complex analysis, we have attempted to focus directly on the potential benefits of low precision in DNN training, aiming to optimize the process without the need to deliberately minimize low-precision noise.

3.1.1 Low precision may improve the performance

This subsection provides the perspective from statistical learning to show that low precision may not hurt the performance of the neural networks, and sometimes even achieve improvement. Let us consider the problem for minimizing a model $\mathbf{w}$ with small population risk $R(\mathbf{w})$:

$$\min_{\mathbf{w}\in\mathbb{R}^d} R(\mathbf{w}) := \mathbb{E}_{\xi\sim\mathcal{D}}\pounds(\mathbf{w};\xi), \tag{1}$$

where ξ denotes the data, and $\mathcal{D}$ is the data distribution, and $\pounds(\mathbf{w};\xi)$ is the loss function satisfying the constraint $\sup_{\mathbf{w}\in\mathbb{R}^d,\xi\sim\mathcal{D}}\{\|\nabla\pounds(\mathbf{w};\xi)\|\}\leqslant U$. For simplicity, $\pounds(\mathbf{w};\xi)$ is assumed to be convex with respect to $\mathbf{w}$. Eq. (1) is an ideal model, characterizing various tasks in the computer vision and artificial intelligence community. Because the distribution $\mathcal{D}$ is usually unknown or very complicated, people use the empirical risk minimization (ERM) $R_S(\mathbf{w})$ surrogate rather than directly solving (1). Given an i.i.d. sampling set $S := \{\xi_1,\xi_2,\ldots,\xi_n\}\sim\mathcal{D}$, ERM is devoted to

$$\min_{\mathbf{w}\in\mathbb{R}^d} R_S(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n}\pounds(\mathbf{w};\xi_i). \tag{2}$$

The main workhorse for Eq. (2) is the stochastic gradient descent (SGD). In [Theorem 5.2, [54]], it has been proved that the generalization error of SGD is

$$R(\mathbf{w}^T) - \min R = O\left(\frac{1+U^2}{\sqrt{T}}\right), \tag{3}$$

where $\mathbf{w}^T$ is the output of SGD after T iterations. From Eq. (3), we can find that the upper bound is related to U: a smaller U indicates a better generalization ability. Recalling that U is the upper bound of the norm of the gradients, that means smaller gradients yield better generalization. It is noteworthy that the gradient here is in the context of full precision training, directly reflecting the change in weights used in consecutive iterations. However, when in the context of low-precision training, the gradients are further approximated. Those tiny gradients which account for the majority are thus neglected by approximation. Consequently, when employing low-precision approximation, we use a gradient with smaller norms to some extent [1)].

Thus, our theoretical analysis indicates that employing low precision not only augments the efficiency of DNN training but also presents the opportunity to attain DNN models with enhanced performance.

3.1.2 Low precision has a similar effect as regularization

DNNs used in practice are over-parametrized to the extent that they can easily fit random labels to the data, but models with increasing capacity will result in overfitting to the training data [55,56]. Regularization is a commonly used method to mitigate overfitting by reducing the variance of over-parameterized model [57], as explained in Fig. 1(a). Similarly, low precision also restricts the model's representation. Therefore, we hypothesize that low precision can be employed to reduce the variance, serving as a form of regularization technique in DNN training.

We verify this phenomenon through extensive experiments, and take the ResNet-38 on CIFAR-10 as an example in Fig. 1(b). We show the variances of layers trained with three different precision configurations: The horizontal axis is the abbreviations for layers, e.g., G1B1C1 represents the 1st conv layer of the 1st block in the 1st group (the structural analysis of models is given in Appendix 1); The vertical axis represents the variance of layers (left, histogram), and the training bit-width of layers (right, dashed line). The blue configuration is the baseline where the entire model is trained using 8-bit, while the orange and green ones use lower precision (4-bit) at some different layers of the model. It can be seen that the variance of layers trained with lower precision is notably smaller than that of the baseline, indicating the regularization effect achieved by utilizing lower precision.

---

1) When training ResNet-38 on CIFAR-10 with 8-bit precision, at iteration 40000, approximately 73.5% of the gradients of layer 5 are neglected due to quantization, and the quantized gradient norm subsequently decreased to 68.1% of its original value.

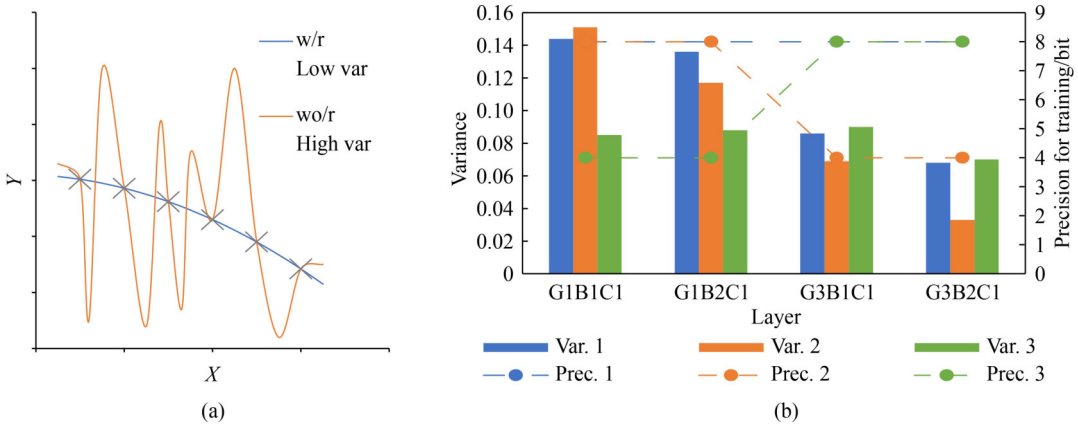


Fig. 1 Low precision reduces the variance in training, similar to regularization. (a) Curve fitted with *vs*. without regularization (w/r *vs*. wo/r); (b) variance (histograms) and training precision (dashed lines) of layers in ResNet-38 on CIFAR-10

Therefore, our experimental observations reveal that low precision in DNN training exhibits similarities to regularization, thereby suggesting the potential benefits in enhancing test accuracy, even without deliberately minimizing quantization error.

3.1.3 Low precision for large capacity layers can help DNN generalization

The representational capacity of DNNs can be analyzed in two dimensions: the amount of data and the precision of each data representation [58]. DNN pruning in training, which can be seen as sparsification of the number of data, is a widely used technique to enhance DNN training [29,59]. Typically, the probability of pruning is proportional to the layer capacity [27,28,60]. Through pruning, DNNs can more effectively acquire sparse knowledge [27], breaking free from local minima [61], and thereby improving generalization [29]. Likewise, low precision can be considered as a means of sparsification in the dimension of the precision of each data. Thus, we hypothesize that low precision for large capacity layers can help DNN generalization.

We use WideResNet-38 as an example to analyze how to determine the relation between bit-width for layer and layer capacity to enhance test accuracy. Figure 2 shows four different training setups based on the layer capacity. The *all 8-bit* (blue) serves as our baseline, using 8-bit for the whole model. The latter three setups assign low precision to both activations and weights: *fixed 6-bit* (red) fixes the bit-width for layers at 6-bit. *8 to 4-bit* (green) assigns lower bit-widths to higher capacity layers, while *4 to 8-bit* (purple) does the opposite. Notably, as the first and last layers require very low computation, the computational costs for the latter three setups are similar.

It can be observed that only the final accuracy of 8 to 4-bit outperform the baseline, despite its lower accuracy compared to other configurations in the initial stage of training. Interestingly, pruning training also exhibits the power to augment final accuracy with a lower test accuracy in the initial stage [27]. These findings highlight the similarity between low precision and pruning, and setting low precision in line with pruning sparsity rates can enhance test accuracy.

To further verify the underlying mechanisms of this improvement, we compare our 8-bit baseline (all 8-bit) with decreases with layer capacity (8 to 4-bit). Figure 3 visualizes the feature embedding of last conv layer in ResNet-110. As expected, the clusters are mixed up together in the baseline model with fixed precision. While decreasing precision with

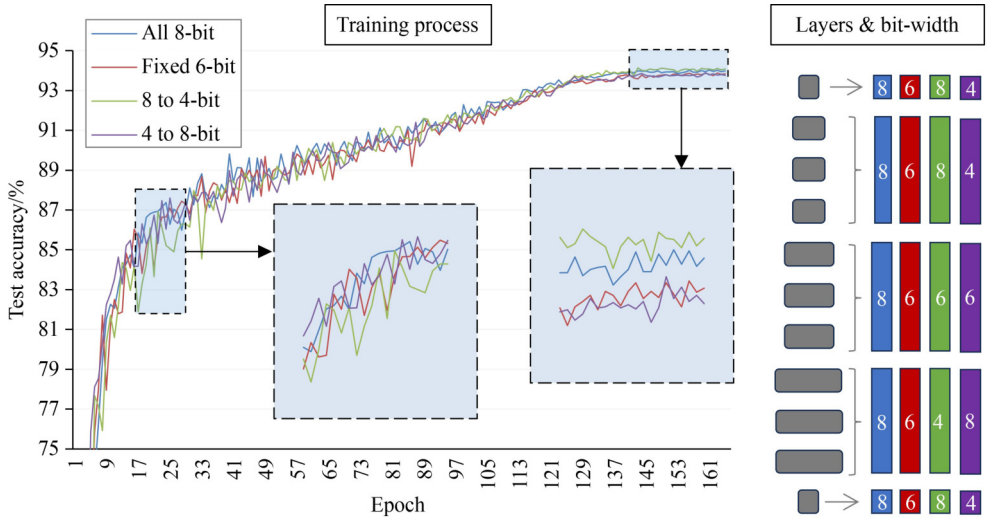
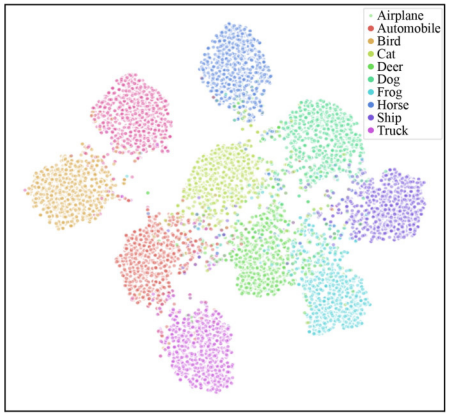
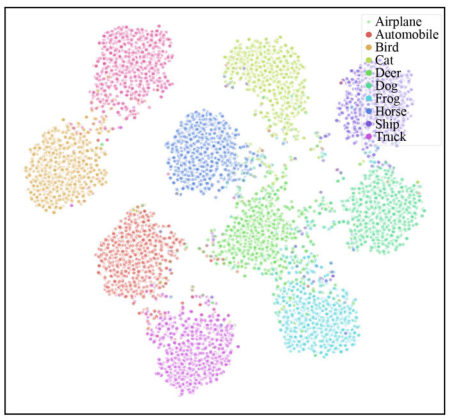


Fig. 2 Test accuracy of WideResNet-38 on CIFAF-10 throughout the training process and the training bit-width for layers



(a)



(b)

Fig. 3 Feature embedding visualization for ResNet-110 using t-SNE on CIFAR-10. (a) Same precision for each layer (8-bit); (b) decreasing precision with layer capacity (8 to 4-bit)

layer capacity can split those clusters with more separable representations (especially the gap between automobile and deer & cat, gap between deer and frog & ship & cat), indicating a lower generalization error over the fixed high-precision baseline.

Thus, we discover that the bit-width selection for each layer in low-precision training shares similarities with the setting of pruning rates, suggesting that assigning higher sparsity (lower bit-width) to layers with larger capacity could potentially enhance the model's generalization capability, thereby achieving higher accuracy.

3.2 Introduction of DPC

The key concept of DPC draws inspiration from our rethinking of low precision as well as recent findings [26] and [29]: The former explores the benefits of low precision temporally and advocates dynamic precision along the training trajectory. The latter demonstrates the strong effectiveness of sparse training in the dimension of parameter count along DNN layers, and identify that appropriate layer-wise sparsity ratios can be an important booster for training a randomly pruned network from scratch. Hence, we have reason to believe that we can leverage lower precision spatially to achieve the win-win in both DNN accuracy and efficiency. Here, we propose DPC, which allocates higher precision to the layers with smaller capacity and lower precision to the layers with larger capacity. Specifically, as shown in Fig. 4, DPC varies the precision across layers within predefined thresholds, as opposed to imposing a static precision across the entire model. This facilitates the application of varying degrees of low-precision regularization to individual layers, thereby potentially enhancing the model's overall performance.

While DPC can be implemented using different layer-wise scheduling methods, here we present as an example an implementation of DPC in a logarithmic manner:

$$B^k = \left\lceil \frac{1}{2}(B_{\max}+B_{\min}) - \frac{\delta}{2}(B_{\max}-B_{\min})\cdot \log_{\max\left(\frac{N_{\max}}{\tilde{N}},\frac{\tilde{N}}{N_{\min}}\right)}\frac{N^k}{\tilde{N}} \right\rfloor, \tag{4}$$

where B^k is the DPC precision for layer k. $B_{\min}$ and $B_{\max}$ are the lower and upper precision bounds, respectively. $\lceil\rfloor$ denotes the rounding operation. N^k is the number of parameters of layer k, $\tilde{N}$ is the number of parameters of intermediate capacity layer, $N_{\min}$ and $N_{\max}$ are the number of parameters of the layer with the smallest capacity and the number of parameters of the layer with the largest capacity, respectively. $\delta \in [0,1]$ is a hyperparameter controlling the interval and range of precision, as shown in Fig. 4. Specifically, Eq. (4) outlines a way to allocate bit-widths for each layer. The bit-widths B^k assigned by Eq. (4) fall between $B_{\max}$ and $B_{\min}$. And $\log_{\max\left(\frac{N_{\max}}{\tilde{N}},\frac{\tilde{N}}{N_{\min}}\right)}\frac{N^k}{\tilde{N}} \in [-1,1]$ uses a logarithmic function to implement the control of the bit-width based on the relative exponential/logarithmic relationship between the layer and the intermediate capacity layer. When N^k is equal to $\tilde{N}$, the bit-width B^k is exactly equal to the intermediate value, i.e., $B^k = \frac{1}{2}(B_{\max}+B_{\min})$. When N^k is less than $\tilde{N}$, then $B^k \in [\frac{1}{2}(B_{\max}+B_{\min}), B_{\max}]$. And when N^k is more than $\tilde{N}$, then $B^k \in [B_{\min}, \frac{1}{2}(B_{\max}+B_{\min})]$. δ further controls the range of bit-width B^k. When the value of δ is smaller, the values of B^k will be more concentrated, while when δ is larger, the upper and lower bounds of B^k will be closer to $B_{\max}$ and $B_{\min}$. We use precision bounds exploration to determine $B_{\max}$ and $B_{\min}$ (see Section 3.3).

In summary, after obtaining the model to be trained, our DPC method first employs precision bounds exploration (Section 3.3) to determine the upper and lower bounds of the training bit-width. Subsequently, based on the model layer capacities, it assigns training bit-widths to different layers according to the approach outlined in Eq. (4), resulting in the DPC model ready for optimized training.

In Section 4.3, we further determine $\delta \in [0.5,1]$, and find DPC is not sensitive to δ in this range. Additionally, we try to use other scheduling functions to make the precision decrease with layer capacity (i.e., not logarithmic schedule in Eq. (4)), and find that the logarithmic function has better performance and stability. We apply DPC to the weights and activations while maintaining a fixed precision for errors and gradients, the latter of which is to ensure the stability of the gradients [23]. More discussion can be found in Section 4.3.

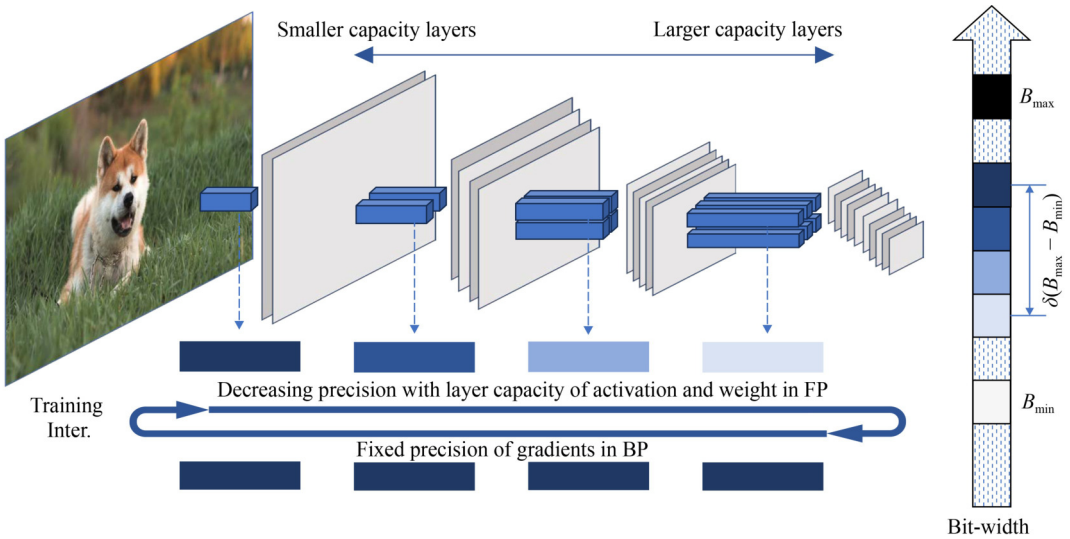


Fig. 4 Decreasing Precision with layer Capacity (DPC), where DPC schedules the precision of weights and activations of layers

3.3 Precision bounds exploration of DPC

In addition to the concept of DPC, the question of how to determine the range of the bit-width has not been addressed, i.e., $B_{\min}$ and $B_{\max}$ in Eq. (4). We propose precision bounds exploration to determine the upper and lower precision bounds of our DPC. Since there are no constraints on the upper precision bound, we use a viable training scheme as a fixed-precision counterpart, setting its bit-width as the upper precision bound, and DPC method further reduces the bit-width used in training. The precision bounds exploration employ a heuristic algorithm to quickly identify the lower precision bound through the application of cosine similarity at a negligible computational expense. The cosine similarity [10] describe the consistency between pre- and post-low-precision quantization (defined in Eq. (5), illustrated in Fig. 5(a), where $Q(\cdot)$ denotes the quantization function and W represents the weight).

$$\cos(\theta) = \frac{W}{Q(W)}. \tag{5}$$

This metric serves as an indicator of the degradation of training performance, exhibiting a pronounced correlation with the ultimate accuracy [41]. For instance, in Fig. 5(b), the accuracy of the final test remains almost unchanged as long as the cosine similarity is maintained at a high level. However, when the cosine similarity begins to decline significantly, the accuracy also tends to decrease correspondingly.

The exploration of precision bounds offers a simple method with negligible computational overhead for selecting the precision bound of DPC, as illustrated in Algorithm 1.

Figure 5(b) offers an intuitive explanation: specifically, we first train the fixed-precision counterpart for a number of iterations (fixed 8-bit, 640 iterations), then extract the weights of the largest capacity layer for analysis (in our DPC method, the largest capacity layer will be assigned the lowest bit-width). Subsequently, we quantize the weights of the layer to a lower bit-width, i.e., 7-bit, and calculate the cosine similarity, which is 0.9998. We then gradually reduce the bit-width and calculate the cosine similarity at each step until the value falls below the threshold (set at 0.95 in our experiments, with a cosine similarity of 0.9463 at 3-bit), indicating a noticeable error due to the low bit-width. We thus take the last bit-width above the threshold (i.e., 4-bit, with a cosine similarity of 0.9876 > 0.95 as shown in Fig. 5(b)) as the lower bound of bit-width.

Algorithm 1 Precision bounds exploration of DPC

Train model with fixed B-bit for first 0.5% iterations, $B_{\max} = B$
Select the largest capacity layer's weight: W
Repeat
  $B \leftarrow B - 1$
  Quantize W to B-bit: $Q(W)$
  $\cos(\theta) = \frac{W}{Q(W)}$
Until $\cos(\theta) < 0.95$
$B_{\min} = B + 1$
Output: upper precision bound $B_{\max}$, lower precision bound $B_{\min}$

As mentioned above, we determined that the upper precision bound of DPC is the bit-width of the fixed-precision counterpart. We also find that DPC demonstrates efficacy across a spectrum of precision ranges, with varying upper bounds $B_{\max}$ (Section 4.3).

4 Experiment

We present the experimental setup in Section 4.1, the overall performance to demonstrate the superiority of DPC in Section 4.2, then comprehensive ablation studies of DPC in Section 4.3, and further analysis to understand the benefits derived from DPC in Section 4.4.

4.1 Experiment setup

Models and datasets. We train ResNet [62] with varying depths and widths and MobileNet-V2 [63,64] on CIFAR-10/100 [65] and ImageNet [66]. For language modeling tasks, we follow [67] for Transformer on WikiText-103 and [68] for LSTM on PTB.

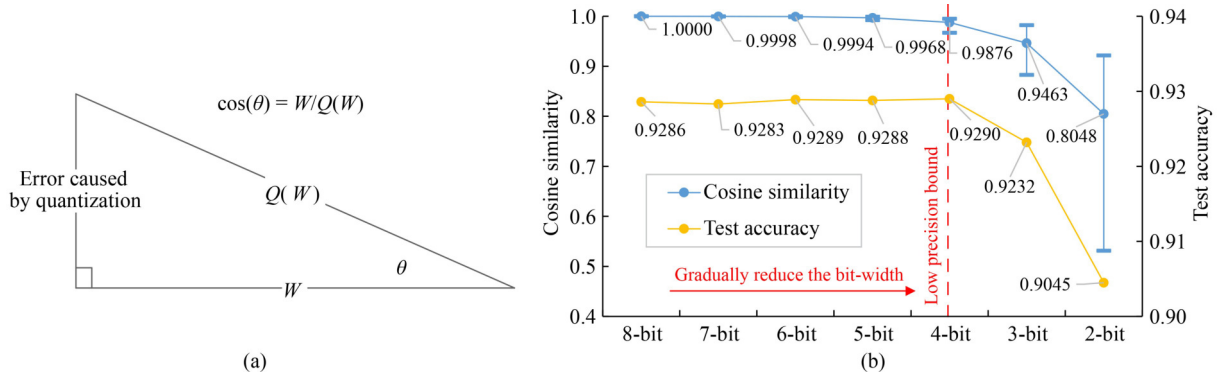


Fig. 5 Determine lower precision bound with the help of cosine similarity. (a) Graphic illustration of cosine similarity; (b) illustrating the exploration of precision bounds for ResNet-38 on CIFAR-10, where the lower precision bound can be set to the switching point

Training settings. Our experiments are based on CPT [26], and on this foundation, our DPC assigns training bit-widths for each layer, thereby training in the manner of DPC. We follow the standard training setting in all the experiments. Specifically, we follow SOTA settings in [69] for CIFAR-10/100 and [62] for ImageNet, respectively. Our experimental data are derived from four independent replications, with the average performance of each method recorded. An analysis of the variance and stability is presented in Section 4.3 and Appendix 2. We choose to employ the GMMLOWP quantization scheme, as detailed in Google's open-source repository, which aligns with the methodologies utilized in CPT [26] and SBM [41]. The explanation of the quantizer is given in Appendix 3.

Precision configurations and baselines. We describe precision configurations in both spatial and temporal dimensions (We use CPT [26] to schedule the training precision in temporal dimension). Specifically, the precision of *spatial* represents the bit-width interval we calculated using the precision bounds exploration of DPC (method proposed in Section 3.3) and Eq. (4) of DPC. For example, *spatial 8 to 4 / temporal 0* donates that the precision ranges from 8-bit to 4-bit as layer capacity increases, and during training, the precision of each layer does not change temporally. And *spatial 8 / temporal 4* donates that the precision is fixed at 8-bit for each layer, while during training, the precision can be reduced by up to 4-bit circularly. In Section 4.2, *temporal −50%* donates the precision of each layer can be cycled down to half the precision of DPC temporally.

We apply our baselines on top of SBM [41], which are listed as follows:

- SBM: The whole DNN is trained with fixed precision, following the framework introduced in [41].
- Pruning: Random pruning techniques as described in [29], based on pre-determined layer-wise sparsities [27].
- CPT: The DNN is trained with dynamic precision, following the method introduced in [26].
- RPC: Precision is randomly assigned regardless of layer capacity.
- CPC: Precision remains constant as layer capacity increases.
- IPC: Precision increments as layer capacity increases.
- DPC: Our proposed method, where precision decreases as layer capacity increases.
- DPC+CPT: Precision for each layer is further cyclically reduced according to CPT, on top of our DPC.

4.2 Overall performance

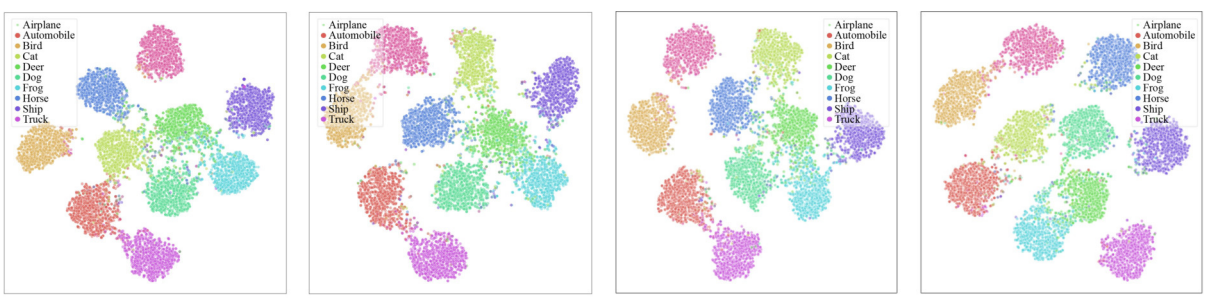
Result on CIFAR. Table 1 presents the experimental results on CIFAR-10/100, and shows that: (1) Our DPC achieves a win-win with both higher accuracy and lower training cost. (2) Compared with other spacial low-precision methods with similar training costs (i.e., RPC, CPC, IPC), DPC achieves higher accuracy consistently, indicating its appropriateness for regularizing DNN training. (3) DPC achieves higher accuracy compared to pruning. In contrast to pruning, low precision offers a gentler way to data representation rather than pruning it directly. (4) In most cases, DPC outperforms the temporal low-precision method CPT. Moreover, CPT requires the entire model to use high precision during inference, whereas DPC can use low precision layer-wise, reducing the memory requirements for hardware, which is critical for deep learning of edge devices. (5) We also test the method using lower precision in both spatial and temporal, DPC+CPT further improves the efficiency while surpassing the accuracy of the 8-bit baseline. This result suggests the potential of using low precision in both spatial and temporal dimensions to enhance DNN training. All the results presented above substantiate the efficacy of DPC, achieving a win-win by enhancing accuracy and reducing computational cost. These findings are also in accordance with our rethinking of the role of low precision in DNN training.

Feature embedding visualization: Figure 6 visualizes the feature embedding of last conv layer in wideResNet-38. We observe that with other methods, certain clusters tend to mix together with relatively small gap. In contrast, DPC splits clusters with wider spacing compared to other methods, indicating DPC helps DNN generalization with more separable feature representations. These results elucidate the reasons behind DPC's ability to enhance accuracy: DPC optimizes the feature representation in DNN training, thereby improving the generalization of the model.

Result on ImageNet. We are also interested in how far we can go with DPC on ImageNet [66], a non-saturated dataset on

Table 1 Test accuracy and computational costs when training ResNet-38/74/110, WideResNet-38 and MobileNet-V2 (α = 0.5/1) on CIFAR-10/100. α is the width multiplier defined in [70]

Model	Method	Prec. (spatial/temporal)	Acc on CIFAR-10/%	Acc on CIFAR-100/%	BitOPs-Sav/%
ResNet-38	SBM	8 / 0	92.86	68.92	–
	RPC	random 4 to 8 / 0	92.59 (−0.27)	68.66 (−0.26)	–
	CPT	8 / 4	92.92 (**+0.06**)	69.62 (−0.30)	30.20
	CPC	6 / 0	92.77 (−0.09)	68.80 (−0.12)	31.08
	IPC	4 to 8 / 0	92.62 (−0.24)	68.01 (−0.91)	30.11
	Pruning	8 / 0	92.57 (−0.29)	68.55 (−0.37)	29.90
	DPC	8 to 4 / 0	92.90 (+0.04)	69.23 (**+0.32**)	30.53
	DPC+CPT	8 to 5 / 2	92.98 (**+0.12**)	70.51 (**+1.59**)	**44.19**
WResNet-38	SBM	8 / 0	93.90	73.39	–
	RPC	random 4 to 8 / 0	94.09 (+0.19)	73.39 (0.00)	–
	CPT	8 / 4	94.02 (+0.12)	73.40 (+0.01)	30.20
	CPC	6 / 0	94.06 (+0.16)	73.66 (+0.27)	31.17
	IPC	4 to 8 / 0	93.91 (+0.01)	73.23 (−0.16)	30.19
	Pruning	8 / 0	94.06 (+0.17)	73.41 (+0.02)	29.90
	DPC	8 to 4 / 0	94.38 (**+0.48**)	73.93 (**+0.54**)	30.61
	DPC+CPT	8 to 5 / 2	94.32 (+0.42)	74.06 (**+0.67**)	**44.30**
ResNet-74	SBM	8 / 0	93.39	70.67	–
	CPT	8 / 4	93.67 (**+0.28**)	71.19 (+0.52)	30.20
	CPC	6 / 0	93.31 (−0.08)	71.16 (+0.49)	31.17
	IPC	4 to 8 / 0	92.97 (−0.42)	70.55 (−0.13)	29.98
	Pruning	8 / 0	93.20 (−0.19)	70.56 (−0.11)	29.90
	DPC	8 to 4 / 0	93.65 (+0.26)	71.35 (**+0.68**)	30.72
	DPC+CPT	8 to 5 / 2	93.76 (**+0.37**)	72.05 (**+1.38**)	**44.42**
ResNet-110	SBM	8 / 0	93.44	71.52	–
	CPT	8 / 4	93.55 (+0.11)	71.73 (+0.21)	30.20
	CPC	6 / 0	93.38 (−0.06)	72.00 (+0.48)	31.20
	IPC	4 to 8 / 0	92.84 (−0.60)	71.35 (−0.16)	29.94
	DPC	8 to 4 / 0	93.69 (**+0.25**)	72.16 (**+0.64**)	30.78
	DPC+CPT	8 to 5 / 2	94.39 (+0.38)	75.46 (**+0.88**)	**44.53**
MN-V2 α = 0.5	SBM	8 / 0	92.83	72.07	–
	CPT	8 / 4	92.80 (−0.03)	71.95 (−0.12)	22.55
	CPC	6 / 0	92.83 (0.00)	71.98 (**-0.09**)	23.35
	IPC	5 to 8 / 0	92.66 (−0.17)	71.40 (−0.67)	20.96
	DPC	8 to 5 / 0	92.89 (**+0.06**)	71.76 (−0.31)	23.87
	DPC+CPT	8 to 6 / 2	93.02 (**+0.19**)	72.46 (**+0.39**)	**34.04**
MN-V2 α = 1	SBM	8 / 0	93.93	75.10	–
	CPT	8 / 4	93.93 (0.00)	74.82 (−0.28)	22.83
	CPC	6 / 0	93.92 (−0.01)	75.17 (+0.07)	23.62
	IPC	5 to 8 / 0	93.80 (−0.13)	74.66 (−0.43)	21.19
	DPC	8 to 5 / 0	93.95 (**+0.02**)	75.43 (**+0.33**)	24.14
	DPC+CPT	8 to 6 / 2	93.92 (−0.01)	75.31 (+0.21)	**34.45**



(a) (b) (c) (d)

Fig. 6 Feature embedding visualization for wideResNet-38 using t-SNE on CIFAR-10. (a) SBM; (b) CPT; (c) IPC; (d) DPC

which DNNs are less over-parameterized and are hard to be compressed [29]. As shown in Table 2, all other methods trade off efficiency with accuracy, whereas DPC achieves comparable accuracy with higher efficiency, and sometimes even achieves higher accuracy. DPC+CPT further improves efficiency, but the excessively low precision leads to training underfitting, resulting in lower accuracy compared to the 8-bit baseline. Nevertheless, when compared to the SBM with

Table 2 Test accuracy and computational costs for the ResNet-34/50 and MobileNet-V2 on ImageNet

Model	Method	Prec. (spatial/temporal)	Acc/%	BitOPs-Sav/%
ResNet-34	SBM	8 / 0	73.94	–
	CPT	8 / 4	73.72 (−0.22)	29.77
	CPC	6 / 0	73.76 (−0.18)	30.80
	IPC	5 to 8 / 0	73.71 (−0.23)	30.77
	DPC	8 to 5 / 0	74.02 (**+0.08**)	30.30
	SBM	6 / 0	72.82	43.85
	DPC+CPT	8 to 5 / 2	73.56 (**+0.74**)	42.88
ResNet-50	SBM	8 / 0	76.32	–
	CPT	8 / 4	76.30 (−0.02)	29.18
	CPC	6 / 0	76.26 (−0.06)	16.34
	IPC	5 to 8 / 0	76.14 (−0.18)	16.24
	DPC	8 to 5 / 0	76.53 (**+0.22**)	23.19
	SBM	6 / 0	75.91	35.33
	DPC+CPT	8 to 5 / 2	76.25 (**+0.34**)	36.91
MN-V2 $\alpha = 2$	SBM	8 / 0	70.02	–
	CPT	8 / 4	69.63 (−0.40)	22.17
	IPC	5 to 8 / 0	69.50 (−0.52)	22.99
	DPC	8 to 5 / 0	69.75 (**−0.17**)	22.32
	SBM	6 / 0	68.01	33.65
	DPC+CPT	8 to 6 / 2	68.43 (**+0.33**)	32.41

similar training costs, DPC+CPT still exhibits higher accuracy, indicating the potential improvement in training with different precision in both spatial and temporal. The results indicate that, despite the increased complexity on the ImageNet, DPC is still capable of optimizing both accuracy and computational training costs.

DPC on full-precision basis: We also evaluate DPC's performance when training based on full precision. As presented in Table 3, DPC achieves a win-win in both efficiency and accuracy in most cases, even on ImageNet. DPC+CPT further improves efficiency by approximately half while maintaining comparable test accuracy (higher than other methods with similar efficiency). Compared with Table 2, DPC achieves higher efficiency while maintaining comparable accuracy. The results demonstrate that DPC is equally effective when applied to a full-precision foundation. And the results is consistent with our insight that low-precision behaves like de-redundancy: Full precision has stronger data representation, there is more room for low-precision optimization.

Table 3 Test accuracy and computational costs on 32-bit based training on ImageNet

Model	Method	Prec. (spatial/temporal)	Acc/%	BitOPs-Sav/%
ResNet-18	Full prec.	32 / 0	70.64	–
	CPT	32 / 24	70.67 (+0.03)	39.01
	DPC	32 to 8 / 0	70.68 (**+0.04**)	39.51
	SBM	22 / 0	70.53	52.53
	DPC+CPT	32 to 8 / −50%	70.53 (0.00)	55.08
ResNet-34	Full prec.	32 / 0	74.13	–
	CPT	32 / 24	74.14 (+0.01)	39.01
	DPC	32 to 8 / 0	74.15 (**+0.02**)	43.24
	SBM	22 / 0	73.99	52.53
	DPC+CPT	32 to 8 / −50%	74.08 (**+0.09**)	58.53
ResNet-50	Full prec.	32 / 0	76.59	–
	DPC	32 to 10 / 0	76.63 (**+0.04**)	44.37
	SBM	22 / 0	76.48	52.53
	DPC+CPT	32 to 10 / −50%	76.57 (**+0.11**)	59.82
MN-V2 $\alpha = 2$	Full prec.	32 / 0	70.51	–
	DPC	32 to 12 / 0	70.51 (0.00)	37.68
	SBM	22 / 0	70.33	52.53
	DPC+CPT	32 to 12 / −50%	70.53 (**+0.20**)	51.49

Result on WikiText-103 and PTB. Table 4 shows that (1) DPC achieves a win-win in both efficiency and model performance (perplexity, the lower the better), and (2) language modeling models/tasks are more sensitive to quantization, as it always adapts to a larger lower precision bound (consistent with observations [26,71]), due to the layers with similar capacities of natural language processing models. The results indicate that, while the performance gains may not be as pronounced in natural language processing models, DPC still manages to bring about optimizations in training.

DPC on VGG and AlexNet. Both ResNet and MobileNet-V2 use a well-designed structure in their blocks, which proved to have stronger representation capacity with similar amount of parameters. In Table 5, we train AlexNet [72] and VGG [73] on CIFAR-10/100 to observe the effect of DPC on DNNs without special structural design. Table 5 reveals the following insights: (1) Based on 8-bit, achieving higher accuracy with DPC becomes challenging. This suggests that relative to ResNet and MobileNet-V2, AlexNet and VGG-16 possess comparatively limited representation capabilities, potentially leading to some degree of underfitting with low precision. (2) Based on 16-bit, DPC outperforms the baseline. The increased precision at this level introduces a certain redundancy in representation capacity, enabling DPC to optimize effectively. (3) DPC remains effective in scenarios involving the combined use of convolutional and fully connected layers (in cases like AlexNet and VGG-16). The observations presented above validate the efficacy of DPC

Table 4 Test accuracy and computational cost of (1) Transformer on WikiText-103 and (2) 2-LSTM on PTB

Model/Dataset	Method	Prec. (spatial/temporal)	Perplexity	BitOPs-Sav/%
Transformer WikiText-103	SBM	8 / 8	31.81	–
	DPC	8 to 5 / 8	**30.44** (**−1.37**)	**23.84**
2-LSTM PTB	SBM	8 / 8	97.11	–
	DPC	8 to 6 / 8	**96.59** (**−0.52**)	**16.21**

Table 5 Test accuracy for training VGG-16 and AlexNet on CIFAR-10/100

Model	Method	Prec.	Acc on CIFAR-10/%	Acc on CIFAR-100/%
AlexNet	SBM	all 8-bit	85.81	59.07
	IPC	4 to 8-bit	85.52 (−0.29)	57.96 (−1.09)
	DPC	8 to 4-bit	85.96 (+0.15)	58.80 (−0.26)
	SBM	all 16-bit	86.21	59.74
	DPC	16 to 4-bit	86.31 (+0.10)	59.89 (+0.15)
VGG-16	SBM	all 8-bit	89.30	62.33
	IPC	4 to 8-bit	88.37 (−0.92)	60.77 (−1.56)
	DPC	8 to 4-bit	89.23 (−0.07)	62.14 (−0.19)
	SBM	all 16-bit	89.35	62.43
	DPC	16 to 4-bit	89.39 (+0.04)	62.60 (+0.17)

across various types of layers, concurrently aligning with the role of low precision in reducing redundancy.

4.3 Ablation studies of DPC

DPC with different precision ranges. We evaluate the performance of DPC across a wide range of upper precision bounds, which correspond to a different target efficiency. Figure 7 illustrates the test accuracy, training cost, and variance (size of circles) from four replicates. We can see that (1) regardless of the upper precision bounds, when compared to the fixed-precision counterpart, DPC consistently achieves a win-win with both higher accuracy and lower training cost, and (2) DPC even reduces accuracy variance, thereby aligning more closely with the practical objective of efficient training.

DPC with different precision intervals. In Eq. (4), different value of δ correspond to distinct precision intervals. We conduct experiments to evaluate the influence on training. As shown in the Table 6, when δ is too small (i.e., $\delta = 0.2$), the precision for layers becomes indistinguishable, leading to degradation of DPC performance. However, within a reasonable interval range ($\delta \in [0.5, 1]$), optimized accuracy can be achieved. Therefore, we set $\delta \in [0.5, 1]$.

DPC with different precision schedules. We evaluate DPC using other precision schedule patterns in addition to the logarithmic one, including a power schedule and a sine schedule. We exclude the linear scheduling based on capacity, as the capacities across layers often exhibit multiplicative relationships. Here, we show several possible scheduling patterns. We employ the ratio between layer capacities as a variable for scheduling and explore a triangular schedule:

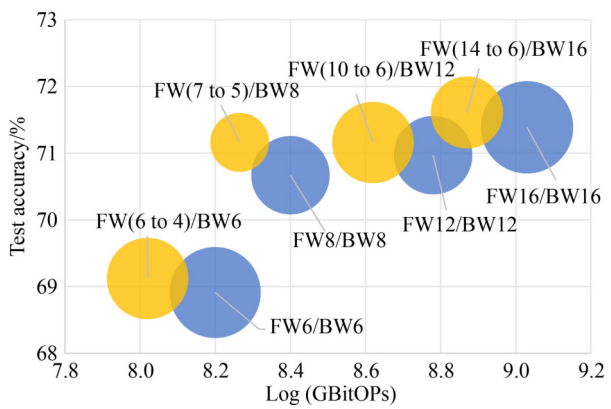


Fig. 7 Training ResNet-74 on CIFAR-100 with DPC and its fixed-precision counterpart. FW/BW denotes precision for weights and activations/errors and gradients, respectively

Table 6 DPC with different precision intervals and precision ranges for ResNet/WResNet-38 on CIFAR-100

Model	Value of δ	Prec.	Acc/%	Prec.	Acc/%
ResNet-38	0.2	6 to 6	68.80	8 to 8	68.92
	0.5	7 to 5	69.01	10 to 6	68.97
	1	8 to 4	**69.23**	12 to 4	**68.98**
WResNet-38	0.2	6 to 6	73.66	8 to 8	73.39
	0.5	7 to 5	**73.93**	10 to 6	**74.33**
	1	8 to 4	**73.93**	12 to 4	74.21

$$B^k = \left\lfloor \frac{1}{2}(B_{\min} + B_{\max}) - \frac{1}{2}(B_{\max} - B_{\min}) \cdot \sin\left[\left(\frac{N^k - N_{\min}}{N_{\max} - N_{\min}} - \frac{1}{2}\right)\pi\right] \right\rfloor, \quad (6)$$

and a power schedule:

$$B^k = \left\lfloor B_{\min} + (B_{\max} - B_{\min}) \cdot \left[1 - \left(\frac{N^k - N_{\min}}{N_{\max} - N_{\min}}\right)^a\right] \right\rfloor. \quad (7)$$

In the power schedule, when a is larger than 1, the precision for layers will change drastically, degrading the final accuracy. Therefore, we set a to 0.5 for scheduling.

Table 7 shows that logarithmic outperform other schedule patterns in terms of both accuracy and stability. Therefore, we chose the logarithmic schedule for DPC.

We suspect that the logarithmic function performs better because of the exponential-logarithmic relationship between bit-width and representation capabilities of data. We leave how to determine the optimal pattern for a given model and task as a future work.

DPC on top of gradients and errors. We opt not to implement DPC on gradients and errors, primarily due to two considerations. Firstly, the instability associated with low-precision gradients and errors, as reported in [23], may impede the model's convergence. Secondly, the precision requirements for gradients and errors typically exceed those for weights and activations, which suggests that the efficiency gains from applying DPC to these components are marginal, as indicated in [26].

To substantiate this decision, we present the outcomes of experiments where DPC was applied to gradients and errors alongside weights and activations. As depicted in Table 8, the application of DPC to gradients and errors offers negligible improvements in accuracy and efficiency when compared to a fixed precision baseline. Consequently, we have chosen to maintain a fixed precision for gradients and errors in all subsequent experimental evaluations.

4.4 Further analysis of DPC

DPC alleviates overfitting during training. When the model size reaches a certain threshold, the test accuracy does not improve with the increased model size due to overfitting. As shown in Table 9, deploying a model with excessive parameters leads to a reduction in test accuracy. Conversely, DPC allows for scaling up the size of models while further improving accuracy without overfitting.

Table 7 DPC with different layer-wise precision schedules for ResNet-38/74 and MobileNet-V2 on CIFAR-100

Schedule	ResNet-38		ResNet-74		MobileNet-V2	
	Acc/%	Var.	Acc./%	Var.	Acc./%	Var.
Logarithmic	69.01	**0.23**	**71.61**	**0.22**	**75.64**	**0.15**
Power	**69.32**	0.32	71.58	0.33	75.01	0.21
Sine	68.89	0.28	71.21	0.38	74.54	0.58

Table 8 Training ResNet-38 and WideResNet-38 on CIFAR-10 with DPC on top of gradient (G) and error (E), where A and W represent activation and weight, respectively

Model	Prec. (W and A/G and E)		Acc/%
	W and A	G and E	
ResNet-38	Fixed 8	Fixed 8	92.86
	Fixed 8	8 to 4	91.77
	Fixed 8	10 to 6	91.97
	Fixed 8	12 to 8	92.84
	8 to 4	8 to 4	91.80
	4 to 8	4 to 8	90.92
WResNet-38	Fixed 8	Fixed 8	93.90
	Fixed 8	8 to 4	92.94
	Fixed 8	12 to 8	93.82
	8 to 4	8 to 4	92.87
	4 to 8	4 to 8	91.04

Table 9 Test accuracy on CIFAR-10/100 without/with DPC

Model	Acc without DPC/%		Acc **with DPC**/%	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
ResNet-74	93.39	70.67	93.65	71.35
WResNet-74	94.01 (+)	74.58 (+)	94.14 (+)	74.78 (+)
WResNet-110	92.62 (−)	74.42 (−)	**94.16 (+)**	**75.30 (+)**

The primary factor contributing to low-precision optimization. Recent findings shows the differences in training between layers from different depths [74,75]. Therefore, it is important to determine whether the depth of the layer or the layer capacity represents the main factor influencing low-precision optimization. To answer this question, we introduced UniformResNet for analysis. This model adopts the same architecture as ResNet, with similar overall computations, but with equal capacity for each layer (details in Appendix 1). Here, we use CPD/IPD/DPD to represent Constant/Increase/Decrease Precision with Depth, respectively.

As shown in Table 10, CPD achieves better results, and DPD is better than IPD. Thus, it seems that the depth of the layer and the layer capacity jointly determine the matching precision, and the latter is dominant. Meanwhile, the accuracy of UniformResNet is far inferior to that of original ResNet with similar computation, which shows the superiority of pyramid architecture of original network [62,63]. Under the reasonable structure, depth and capacity of layers are consistent from the precision perspective.

DPC alleviates the vanishing gradients. We observe that DPC alleviates the degradation caused by vanishing gradients. Typically, the initial layers are challenging to train due to small gradients [76]. In Fig. 8, we present the weight gradients of layers from the initial shallow-depth group (e.g., B1C1: 1st convolution in 1st block) of ResNet-38 trained under different precision configurations. At epoch 100 (Fig. 8(a)), gradient magnitudes of SBM, IPC and DPC exhibit minor discrepancies, with gradients still not considerably diminished. However, by epoch 150 (Fig. 8(b)), the gradients have substantially diminished, notably showcasing a marked increase in the gradient of DPC in comparison to those of SBM and IPC.

This phenomenon highlights the fact that DPC enables shallow-depth layers to be adequately updated in the final stage, thus mitigating the issue of vanishing gradients.

The underlying cause of this phenomenon can be explained by the analysis in [19,52] (in Section 5).

5 Discussion

5.1 Scale of models

From all the experimental results, we observe that models with larger scale (deeper/wider) are more friendly to low precision (achieving higher accuracy with lower computational cost), which shares similarities with recent finding in pruning [29]. On one hand, large models have more parameters, thereby providing more opportunities to identify valuable features using low-precision techniques. Conversely, models with intensive structure necessitate a more fine-grained representation and more careful adjustment for each parameter, rendering them more sensitive to changes in precision. The conjecture is somewhat similar to lottery ticket hypothesis proposed in [60]. On the other hand, the use of low precision is also capable of suppressing interfering features, thereby facilitating the learning of crucial features.

5.2 Formulation of low-precision training

Recent studies [19,52] investigate how does low-precision quantization affect DNN training by analyzing gradients variance. In [Eq. (7) in [52]], the gradient variance for Quantization Aware Training (QAT) can be written as

$$\mathrm{Var}[\widehat{\nabla}_{\Theta^{(l)}}] = \mathrm{Var}[\nabla_{\Theta^{(l)}}] + \sum_{m=l}^{L} \mathbb{E}\left[\mathrm{Var}\left[\mathrm{G}_{\Theta}^{(l\sim m)}\left(\widehat{\nabla}_{H^{(m)}}, \widehat{C}^{(m)}\right)\middle|\widehat{\nabla}_{H^{(m)}}\right]\right]. \quad (8)$$

Table 10 Test accuracy for UniformResNet on CIFAR-10/100

Model	Method	Prec.	CIFAR-10		CIFAR-100	
			Acc/%	Impro/%	Acc/%	Impro/%
ResNet38	SBM	8 / 0	92.86	–	68.92	–
Uniform ResNet38	SBM	8 / 0	90.83	(−2.03)	60.69	(−8.23)
	CPD	6 / 0	90.88	+0.05(−2.03)	60.82	+0.13(−8.23)
	IPD	4 to 8/0	90.54	−0.29(−2.03)	60.03	−0.66(−8.23)
	DPD	8 to 4/0	90.71	−0.12(−2.03)	60.33	−0.36(−8.23)
ResNet74	SBM	8 / 0	93.39	–	70.67	–
Uniform ResNet74	SBM	8 / 0	91.84	(−1.55)	63.68	(−6.99)
	CPD	6 / 0	91.96	+0.12(−1.55)	63.80	+0.12(−6.99)
	IPD	4 to 8/0	91.73	−0.11(−1.55)	63.47	−0.21(−6.99)
	DPD	8 to 4/0	91.82	−0.02(−1.55)	63.37	−0.31(−6.99)

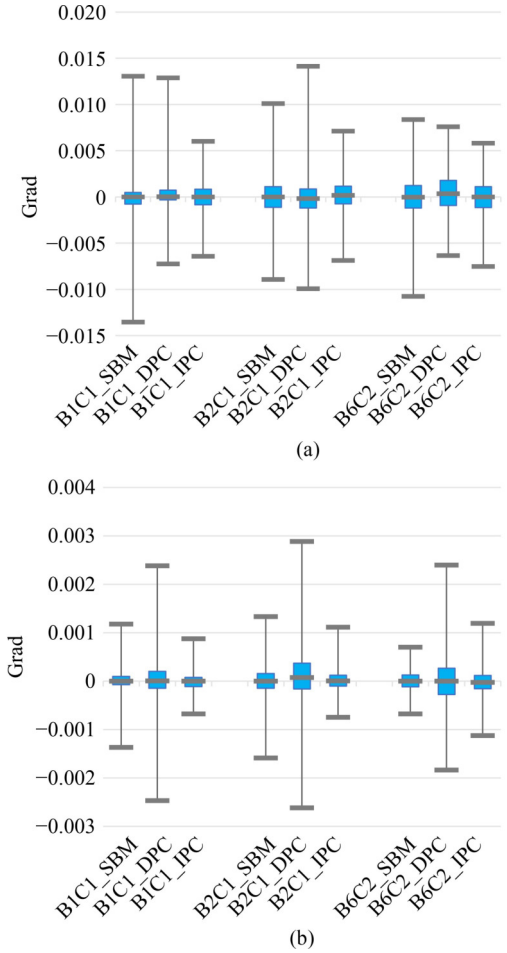


Fig. 8 DPC alleviates the vanishing gradients. The weight gradients of three shallow-depth layers of ResNet-38 trained with SBM, DPC, and IPC on CIFAR-10, respectively. (a) Data distributions of weight gradients (epoch 100); (b) data distributions of weight gradients (epoch 150)

The first term in is just the full precision gradient variance, and it accounts for the minibatch sampling. All the rest terms account for the noise of utilizing compressed context. Specifically, the term with $\mathrm{G}_{\Theta}^{(l\sim m)}\left(\cdot,\widehat{C}^{(m)}\right)$ is the variance introduced by utilizing the compressed context $\widehat{C}^{(m)}$.

And the gradient variance for Fully Quantized Training (FQT) can be written as [Eq. (7) in [19]]:

$$\mathrm{Var}[\widehat{\nabla}_{\Theta}] = \mathrm{Var}[\nabla_{\Theta}] + \sum_{l=1}^{L}\mathbb{E}\left[\sum_{k=1}^{l}\mathrm{Var}\left[\mathrm{vec}(Q_b(\widehat{\nabla}_{H^{(l)}}))\gamma^{(k,l)}\middle|\widehat{\nabla}_{H^{(l)}}\right]\right], \quad (9)$$

where the first term $\mathrm{Var}[\widehat{\nabla}_{\Theta}]$ is the variance of the QAT gradient. All the remaining variance terms come from gradient quantization, where each term $\mathrm{Var}\left[\mathrm{vec}(Q_b(\widehat{\nabla}_{H^{(l)}}))\gamma^{(k,l)}\middle|\widehat{\nabla}_{H^{(l)}}\right]$ is the variance imposed by the quantizer $Q_b(\widehat{\nabla}_{H^{(l)}})$ on layer l to the gradient $\widehat{\nabla}_{\Theta^{(k)}}$ on layer k. These formulations highlight that the distribution of each layer is not solely determined by its own; rather, the variance of each layer is also influenced by the quantization of other layers. During training, the entire model adjusts collectively to accommodate its low-precision parts.

From our analysis of the model structure in Appendix 1, it can be found that capacities of initial layers tend to be small. Consequently, under our DPC, higher precision is employed in these initial layers, while lower precision for deep layers. According to Eqs. (8) and (9), low precision introduces additional variance, and the variance of the initial layer compounds with the variance introduced by subsequent layers employing low precision. In the final training stages, gradients concentrate near zero [3,13]. Hence, augmenting variance under such circumstances serves to increase gradients, thus mitigating the issue of vanishing gradients.

6 Conclusion and future work

We systematically investigate the benefits of employing low-precision in DNN training, which can be proven to (i) optimize DNNs performance, serve as a form of (ii) regularization and (iii) layer-wise de-redundancy. Building upon these explorations, we introduce a simple yet powerful technique – DPC. Extensive experiments validate that comparable or even superior accuracy can be achieved even without deliberately minimizing quantization errors. Our technique and experimental results contribute to a deeper understanding of low-precision training, offering a new perspective for simultaneously optimizing efficiency and model performance.

Our future works will focus on identifying more theoretical grounds and further analyze low-precision methods across both spatial and temporal dimensions. Additionally, Large-scale Language Models (LLMs) demonstrate remarkable capabilities but often require substantial training costs. Current widely-used approaches focus on compressing the model Post-Training Quantization (PTQ) [77], with limited application of low-precision techniques during the pre-training phase. Therefore, exploring how to integrate the perspectives discussed in this paper with low-precision methods during the training of LLMs to enhance efficiency and potentially improve model performance is a promising research direction. Considering the substantial cost associated with pre-training LLMs, training LLMs directly with low precision is challenging. Therefore, it may be beneficial to initially attempt to optimize the fine-tuning of LLMs by combining PTQ with fine-tuning [78], and further distinguishing the bit-widths of different layers.

Acknowledgements This work was supported by the National Key R&D Program of China (Grant No. 2021YFB0301200) and the National Natural Science Foundation of China (Grant No. 62025208).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from

the copyright holder.

To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendixes

Appendix 1 — The structural analysis of models

ResNet and MobileNet-V2. We utilize ResNet-34/50 and MobileNet-V2 ($\alpha = 1$) on ImageNet as an example to analyze the structure of our models in our experiment. As illustrated in Table A1, it's shown that with increasing depth, feature maps consistently decrease in size. Additionally, apart from the first and last layers, the convolutional kernel sizes expand, becoming thicker and possessing more channels. This phenomenon stands as a common trait within CNN architectures [62,63,70,73,79]. This feature balances the computation at different depths and is consistent with our cognitive characteristics: Shallow layers have small receptive fields and require large feature maps to represent overall information, whereas deep layers have large receptive fields and can use small feature maps to represent overall information. Moreover, deep features are combinations of shallow features, requiring more channels to represent. As seen from Table A1, MobileNet-V2 has fewer channels than ResNet and is a "thinner" model.

Figure A1 illustrates the structure of three different blocks. BasicBlock and BottleNeck have similar numbers of parameters, and BottleNeck has more layers. In contrast, InverseBottleNeck is more intensive, splitting the large convolution kernel into depth-wise convolution and point-wise convolution. Especially for depth-wise convolution, only a few parameters are used to complete the feature extraction, requiring each parameter to be carefully adjusted. Thus, MobileNet-V2 is more sensitive to the low-precision quantization.

WideResNet and UniformResNet. To analyze the effects of model scales, we use WideResNet, which doubles the channel, and to analyze the effects of the number of parameters and layer depth, we use uniformResNet, which has the same number of channels in each group (similar computational cost in total). The structural comparison of ResNet, WideResNet, and UniformResNet on CIFAR is presented in Table A2.

Table A1 Structure of ResNet [62] and MobileNet-V2 [63] on ImageNet

ResNet-34/50		MobileNet-V2		Output
Operator	Channel	Operator	Channel	
conv_7*7*3	64	conv_3*3*3	32	112*112
		InverseBottleNeck	24	
		InverseBottleNeck	24	
pool				56*56
BasicBlock/BottleNeck	64	InverseBottleNeck	24	
BasicBlock/BottleNeck	64	InverseBottleNeck	24	
BasicBlock/BottleNeck	64			
BasicBlock/BottleNeck	128	InverseBottleNeck	32	28*28
BasicBlock/BottleNeck	128	InverseBottleNeck	32	
BasicBlock/BottleNeck	128	InverseBottleNeck	32	
BasicBlock/BottleNeck	128			
BasicBlock/BottleNeck	256	InverseBottleNeck	64	14*14
BasicBlock/BottleNeck	256	InverseBottleNeck	64	
BasicBlock/BottleNeck	256	InverseBottleNeck	64	
BasicBlock/BottleNeck	256	InverseBottleNeck	64	
BasicBlock/BottleNeck	256	InverseBottleNeck	96	
BasicBlock/BottleNeck	256	InverseBottleNeck	96	
		InverseBottleNeck	96	
BasicBlock/BottleNeck	512	InverseBottleNeck	160	7*7
BasicBlock/BottleNeck	512	InverseBottleNeck	160	
BasicBlock/BottleNeck	512	InverseBottleNeck	160	
BasicBlock/BottleNeck	512	InverseBottleNeck	320	
pool		conv_1*1*320	1280	1*1
fc	1000	fc	1000	

Appendix 2 — The standard variations

Our experimental results on CIFAR are the average of 4 replicate experiments. Some of the standard variations are shown in Table A3. It can be seen that fluctuations do exist in the experimental results of all methods. However, despite these fluctuations, our DPC achieves higher test accuracy in most cases.

According to the analysis of sparse training, some parameters are basically not modified once they are established [61]. Therefore, low-precision training is susceptible to randomness. We suspect more reasonable updates at some key stages in training may reduce the fluctuations, and we leave this as our future work.

Appendix 3 — Quantization methods

We choose to employ the GMMLOWP quantization scheme, as detailed in Google's open-source repository, which aligns

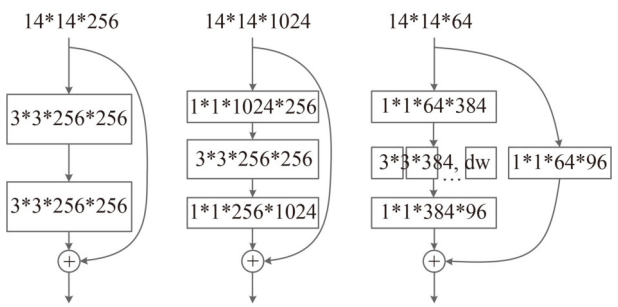


Fig. A1 Three kinds of blocks (on 14*14 feature maps) for ImageNet. Left: a BasicBlock for ResNet-34; Middle: a BottleNeck for ResNet-50; Right: a InverseBottleNeck for MobileNet-V2

Table A2 Structural comparison of ResNet, WideResNet and uniformResNet on CIFAR

Group	Operator	Repeated times	Output	Number of channels		
				ResNet-38/74/110	WideResNet-38/74/110	UniformResNet-38/74/110
	conv_3*3*3	1	32*32	16	32	20
Group 1	BasicBlock	6/12/18	32*32	16	32	20
Group 2	BasicBlock	6/12/18	16*16	32	64	20
Group 3	BasicBlock	6/12/18	8*8	64	128	20
	fc	1	1*1	10 or 100	10 or 100	10 or 100

Table A3 Standard variations of the results on CIFAR-10/100

Method	Standard variations on (CIFAR-10,CIFAR-100)			
	ResNet-38	WResNet-38	ResNet-110	MN-V2
SBM	0.11, 0.24	0.12, 0.17	0.19, 0.27	0.13, 0.30
RPC	0.20, 0.44	0.20, 0.35	–	–
CPC	0.20, 0.34	0.08, 0.30	0.14, 0.29	0.13, 0.14
IPC	0.23, 0.25	0.11, 0.24	0.44, 0.44	0.13, 0.27
*DPC	0.19, 0.32	0.07, 0.25	0.07, 0.33	0.11, 0.19
CPT	0.19, 0.40	0.16, 0.27	0.17, 0.35	0.24, 0.40
*DPC+CPT	0.10, 0.36	0.12, 0.14	0.34, 0.43	0.34, 0.43

with the methodologies utilized in CPT [26] and SBM [41]. For a given input tensor $\mathbf{X}$, clamping range $[v_{\min}, v_{\max}]$, and bit-width B, the output is formulated as follows:

$$scale = \frac{v_{\max} - v_{\min}}{2^B},$$

$$zero - point = \text{round}(\min(\max(-v_{\min}/scale, 0), 2^B)),$$

$$output = \text{round}(x/scale + zero - point).$$

The clamping values are defined as the maximum and minimum of the tensor.

References

1. Du J, Shen M, Du Y. A distributed in-situ CNN inference system for IOT applications. In: Proceedings of the 38th IEEE International Conference on Computer Design (ICCD). 2020, 279–287
2. Yang C, Wu Z, Chee J, De Sa C, Udell M. How low can we go: trading memory for error in low-precision training. In: Proceedings of the 10th International Conference on Learning Representations. 2022
3. Zhu F, Gong R, Yu F, Liu X, Wang Y, Li Z, Yang X, Yan J. Towards unified int8 training for convolutional neural network. In: Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 1966–1976
4. Micikevicius P, Narang S, Alben J, Diamos G, Elsen E, Garcia D, Ginsburg B, Houston M, Kuchaiev O, Venkatesh G, Wu H. Mixed precision training. In: Proceedings of the 6th International Conference on Learning Representations. 2018
5. Zhou S, Wu Y, Ni Z, Zhou X, Wen H, Zou Y. DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. 2016, arXiv preprint arXiv: 1606.06160
6. Yang L, Jin Q. FracBits: mixed precision quantization via fractional bit-widths. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 10612–10620
7. Elthakeb A T, Pilligundla P, Mireshghallah F, Yazdanbakhsh A, Esmaeilzadeh H. ReLeQ: a reinforcement learning approach for automatic deep quantization of neural networks. IEEE Micro, 2020, 40(5): 37−45.
8. Yang H, Duan L, Chen Y, Li H. BSQ: exploring bit-level sparsity for mixed-precision neural network quantization. In: Proceedings of the 9th International Conference on Learning Representations, 2021
9. Ma Z, He J, Qiu J, Cao H, Wang Y, Sun Z, Zheng L, Wang H, Tang S, Zheng T, Lin J, Feng G, Huang Z, Gao J, Zeng A, Zhang J, Zhong R, Shi T, Liu S, Zheng W, Tang J, Yang H, Liu X, Zhai J, Chen W. BaGuaLu: targeting brain scale pretrained models with over 37 million cores. In: Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2022, 192–204
10. Lee S, Park J, Jeon D. Toward efficient low-precision training: data format optimization and hysteresis quantization. In: Proceedings of the 10th International Conference on Learning Representations. 2021
11. Sun X, Wang N, Chen C Y, Ni J M, Agrawal A, Cui X, Venkataramani S, El Maghraoui K, Srinivasan V V, Gopalakrishnan K. Ultra-low precision 4-bit training of deep neural networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 33: 1796–1807
12. Fu Y, You H, Zhao Y, Wang Y, Li C, Gopalakrishnan K, Wang Z, Lin Y. FracTrain: fractionally squeezing bit savings both temporally and spatially for efficient DNN training. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1017
13. Koster U, Webb T J, Wang X, Nassar M, Bansal A K, Constable W H, Elibol O H, Gray S, Hall S, Hornof L, Khosrowshahi A, Kloss C, Pai R J, Rao N. Flexpoint: an adaptive numerical format for efficient training of deep neural networks. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 1740–1750
14. Das D, Mellempudi N, Mudigere D, Kalamkar D, Avancha S, Banerjee K, Sridharan S, Vaidyanathan K, Kaul B, Georganas E, Heinecke A, Dubey P, Corbal J, Shustrov N, Dubtsov R, Fomenko E, Pirogov V. Mixed precision training of convolutional neural networks using integer operations. In: Proceedings of the 6th International Conference on Learning Representations. 2018
15. Fox S, Rasoulinezhad S, Faraone J, Leong P, Leong P. A block minifloat representation for training deep neural networks. In: Proceedings of the 9th International Conference on Learning Representations. 2020
16. Sun X, Choi J, Chen C Y, Wang N, Venkataramani S, Srinivasan V V, Cui X, Zhang W, Gopalakrishnan K. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 441
17. Zhang D, Yang J, Ye D, Hua G. LQ-Nets: learned quantization for highly accurate and compact deep neural networks. In: Proceedings of the 15th European Conference on Computer Vision-ECCV 2018. 2018, 373–390
18. Choi J, Wang Z, Venkataramani S, Chuang P I J, Srinivasan V, Gopalakrishnan K. PACT: parameterized clipping activation for quantized neural networks. In: Proceedings of the 6th International Conference on Learning Representations, 2018
19. Chen J, Gai Y, Yao Z, Mahoney M W, Gonzalez J E. A statistical framework for low-bitwidth training of deep neural networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 75
20. Fu F, Hu Y, He Y, Jiang J, Shao Y, Zhang C, Cui B. Don't waste your bits! Squeeze activations and gradients for deep neural networks via TINYSCRIPT. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 309
21. Wu S, Li G, Chen F, Shi L. Training and inference with integers in deep neural networks. In: Proceedings of the 6th International Conference on Learning Representations, 2018
22. Yang Y, Deng L, Wu S, Yan T, Xie Y, Li G. Training high-performance and large-scale deep neural networks with full 8-bit integers. Neural Networks, 2020, 125: 70−82.

23. Wang N, Choi J, Brand D, Chen C Y, Gopalakrishnan K. Training deep neural networks with 8-bit floating point numbers. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018, 7686–7695
24. Grandvalet Y, Canu S, Boucheron S. Noise injection: theoretical prospects. Neural Computation, 1997, 9(5): 1093−1108.
25. Neelakantan A, Vilnis L, Le Q V, Sutskever I, Kaiser L, Kurach K, Martens J. Adding gradient noise improves learning for very deep networks. In: Proceedings of the 5th International Conference on Learning Representations. 2017
26. Fu Y, Guo H, Li M, Yang X, Ding Y, Chandra V, Lin Y. CPT: efficient deep neural network training via cyclic precision. In: Proceedings of the 9th International Conference on Learning Representations. 2021
27. Mocanu D C, Mocanu E, Stone P, Nguyen P H, Gibescu M, Liotta A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature Communications, 2018, 9(1): 2383.
28. Evci U, Gale T, Menick J, Castro P S, Elsen E. Rigging the lottery: making all tickets winners. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 2943–2952
29. Liu S, Chen T, Chen X, Shen L, Mocanu D C, Wang Z, Pechenizkiy M. The unreasonable effectiveness of random pruning: return of the most naive baseline for sparse training. In: Proceedings of the 10th International Conference on Learning Representations, ICLR 2022. 2022, 21
30. Zhou A, Yao A, Guo Y, Xu L, Chen Y. Incremental network quantization: towards lossless CNNs with low-precision weights. In: Proceedings of the 5th International Conference on Learning Representations, 2017
31. Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: a whitepaper. 2018, arXiv preprint arXiv: 1806.08342
32. Banner R, Nahshan Y, Soudry D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 714
33. Wang K, Liu Z, Lin Y, Lin J, Han S. HAQ: hardware-aware automated quantization with mixed precision. In: Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 8604–8612
34. Guo C, Tang J, Hu W, Leng J, Zhang C, Yang F, Liu Y, Guo M, Zhu Y. OliVe: accelerating large language models via hardware-friendly outlier-victim pair quantization. In: Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023, 3
35. Guo C, Zhang C, Leng J, Liu Z, Yang F, Liu Y, Guo M, Zhu Y. ANT: exploiting adaptive numerical data type for low-bit deep neural network quantization. In: Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). 2022, 1414–1433
36. Onan A, Korukoğlu S, Bulut H. A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. Information Processing & Management, 2017, 53(4): 814−833.
37. Achille A, Rovere M, Soatto S. Critical learning periods in deep networks. In: Proceedings of the 7th International Conference on Learning Representations. 2018
38. Achille A, Soatto S. Emergence of invariance and disentanglement in deep representations. Journal of Machine Learning Research, 2018, 19(1): 1947−1980.
39. Raghu M, Poole B, Kleinberg J, Ganguli S, Sohl Dickstein J. On the expressive power of deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning. 2017, 2847–2854
40. Martinez B, Yang J, Bulat A, Tzimiropoulos G. Training binary neural networks with real-to-binary convolutions. In: Proceedings of the 8th International Conference on Learning Representations. 2020
41. Banner R, Hubara I, Hoffer E, Soudry D. Scalable methods for 8-bit training of neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing systems. 2018, 5151–5159
42. Park E, Yoo S. PROFIT: a novel training method for sub-4-bit MobileNet models. In: Proceedings of the 16th European Conference on Computer Vision. 2020, 430–446
43. Yang G, Zhang T, Kirichenko P, Bai J, Wilson A G, De Sa C. SWALP: stochastic weight averaging in low precision training. In: Proceedings of the 36th International Conference on Machine Learning. 2019, 7015–7024
44. Han R, Si M, Demmel J, You Y. Dynamic scaling for low-precision learning. In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2021, 480–482
45. Feng B, Wang Y, Geng T, Li A, Ding Y. APNN-TC: accelerating arbitrary precision neural networks on ampere GPU tensor cores. In: Proceedings of the SC21: International Conference for High Performance Computing, Networking, Storage and Analysis. 2021, 1–14
46. Knorr F, Thoman P, Fahringer T. ndzip-gpu: efficient lossless compression of scientific floating-point data on GPUs. In: SC21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021, 1–13
47. Savarese P, Yuan X, Li Y, Maire M. Not all bits have equal value: heterogeneous precisions via trainable noise. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 35769–35782
48. Xi H, Li C, Chen J, Zhu J. Training transformers with 4-bit integers. In: Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023). 2023
49. Huang X, Shen Z, Li S, Liu Z, Hu X, Wicaksana J, Xing E, Cheng K T. SDQ: stochastic differentiable quantization with mixed precision. In: Proceedings of the 39th International Conference on Machine Learning. 2022, 9295–9309
50. Chakrabarti A, Moseley B. Backprop with approximate activations for memory-efficient network training. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 218
51. Jin S, Li G, Song S L, Tao D. A novel memory-efficient deep learning training framework via error-bounded lossy compression. In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2021, 485–487
52. Chen J, Zheng L, Yao Z, Wang D, Stoica I, Mahoney M, Gonzalez J. ActNN: reducing training memory footprint via 2-bit activation compressed training. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 1803–1813
53. Zhuang B, Shen C, Tan M, Liu L, Reid I. Towards effective low-bitwidth convolutional neural networks. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, 7920–7928
54. Hardt M, Recht B, Singer Y. Train faster, generalize better: Stability of stochastic gradient descent. In: Proceedings of the 33rd International Conference on Machine Learning. 2016, 1225–1234
55. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 2021, 64(3): 107−115.
56. Neyshabur B, Li Z, Bhojanapalli S, LeCun Y, Srebro N. Towards understanding the role of over-parametrization in generalization of neural networks. 2018, arXiv preprint arXiv: 180512076
57. Poggio T, Torre V, Koch C. Computational vision and regularization theory. In: Fischler M A, Firschein Q, eds. Readings in Computer Vision: Issues, Problem, Principles, and Paradigms. Los Altos: Morgan Kaufmann, 1987, 638–643
58. Dettmers T, Zettlemoyer L. The case for 4-bit precision: *k*-bit inference scaling laws. In: Proceedings of the 40th International Conference on

Machine Learning. 2023, 370
59. Su J, Chen Y, Cai T, Wu T, Gao R, Wang L, Lee J D. Sanity-checking pruning methods: random tickets can win the jackpot. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 33: 1712
60. Frankle J, Carbin M. The lottery ticket hypothesis: finding sparse, trainable neural networks. In: Proceedings of the 7th International Conference on Learning Representations. 2019
61. Mostafa H, Wang X. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In: Proceedings of the 36th International Conference on Machine Learning. 2019, 4646–4655
62. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. 2016, 770–778
63. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L C. MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, 4510–4520
64. Wang Y, Jiang Z, Chen X, Xu P, Zhao Y, Lin Y, Wang Z. E^2-train: training state-of-the-art CNNs with over 80% energy savings. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 462
65. Krizhevsky A. Learning multiple layers of features from tiny images. University of Toronto, 2012
66. Deng J, Dong W, Socher R, Li L J, Li K, Fei-Fei L. ImageNet: a large-scale hierarchical image database. In: Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009, 248–255
67. Baevski A, Auli M. Adaptive input representations for neural language modeling. In: Proceedings of the 7th International Conference on Learning Representations. 2019
68. Merity S, Keskar N S, Socher R. Regularizing and optimizing LSTM language models. In: Proceedings of the 6th International Conference on Learning Representations. 2018
69. Wang X, Yu F, Dou Z Y, Darrell T, Gonzalez J E. SkipNet: learning dynamic routing in convolutional networks. In: Proceedings of the 15th European Conference on Computer Vision-ECCV 2018. 2018, 420–436
70. Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: efficient convolutional neural networks for mobile vision applications. 2017, arXiv preprint arXiv: 1704.04861
71. Hou L, Zhu J, Kwok J, Gao F, Qin T, Liu T Y. Normalization helps training of quantized LSTM. In: Proceedings of the 33rd Neural Information Processing Systems. 2019, 660
72. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6): 84−90.
73. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv preprint arXiv: 1409.1556
74. He C, Li S, Soltanolkotabi M, Avestimehr S. PipeTransformer: automated elastic pipelining for distributed training of transformers. 2021, arXiv preprint arXiv: 2102.03161
75. Raghu M, Gilmer J, Yosinski J, Sohl-Dickstein J. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 6078–6087
76. Nielsen M A. Neural Networks and Deep Learning. San Francisco, CA, USA: Determination Press, 2015
77. Frantar E, Ashkboos S, Hoefler T, Alistarh D. GPTQ: accurate post-training quantization for generative pre-trained transformers. In: Proceedings of the 11th International Conference on Learning Representations. 2023
78. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLORA: efficient finetuning of quantized LLMs. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2024, 441
79. Huang G, Liu Z, Van Der Maaten L, Weinberger K Q. Densely connected convolutional networks. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition. 2017, 2261–2269



Ao Shen received his MS degree in Communication and Information Engineering from National University of Defense Technology (NUDT), China in 2019. He is pursuing his PhD degree at the College of Computer, NUDT, China. His current interests are mainly in optimization for machine learning and high-performance system software.



Zhiquan Lai received his PhD degrees in Computer Science from National University of Defense Technology (NUDT), China in 2015. He is currently an associate professor in the National Key Laboratory for Parallel and Distributed Processing of NUDT, China. He worked as a research assistant at Department of Computer Science, the University of Hong Kong, China from Oct. 2012 to Oct. 2013. His current research interests include high-performance system software, distributed machine learning.



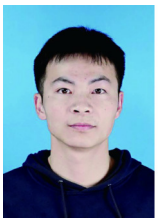
Tao Sun received his PhD degree from National University of Defense Technology (NUDT), China in 2018. Currently, he is an assistant professor with National Laboratory for Parallel and Distributed Processing, NUDT, China. His research interests include optimization for machine learning.



Shengwei Li received the BS degree from Nanjing University, China in 2017, and the MS degree in computer science from Stony Brook University, USA in 2020. He is pursuing his PhD degree at the College of Computer, NUDT, China. His research interests include high-performance computing and distributed machine learning systems.



Keshi Ge received his BS degree from the Department of Computer Science and Technology, Xi'an Jiaotong University, China in 2015, and his PhD and MS degrees from the College of Computer, National University of Defense Technology (NUDT), China in 2022 and 2017, respectively. He worked as a visiting PhD student at the Department of Electrical and Computer Engineering, University of Alberta, Canada from Nov. 2019 to Aug. 2020. He is currently an Assistant Professor with NUDT. His research interests include high-performance computing and distributed machine learning systems.



Weijie Liu received his Bachelor degree in computer science from Nankai University, China in 2020 and his MS degree from the College of Computer, National University of Defense Technology (NUDT), China in 2022. He is pursuing his PhD degree at the College of Computer, NUDT, China. His current interests are mainly in optimization techniques related to large-scale model training.



Dongsheng Li received his PhD degree in computer science and technology from National University of Defense Technology (NUDT), China in 2005. He is currently a professor and doctoral supervisor in the College of Computer at NUDT, China. He was awarded the Chinese National Excellent Doctoral Dissertation and the National Science Fund for Excellent Young Scholars. His research interests include distributed systems, cloud computing and big data processing.