Journal of King Saud University – Computer and Information Sciences 35 (2023) 101656

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Layered mixed-precision training: A new training method for large-scale AI models

Hao Li, Yuzhu Wang *, Yan Hong, Fei Li, Xiaohui Ji

School of Information Engineering, China University of Geosciences, No. 29, Xueyuan Road, Beijing 100083, China

ARTICLE INFO

Article history:
Received 26 May 2023
Revised 6 July 2023
Accepted 12 July 2023
Available online 24 July 2023

Keywords:
Deep learning
Mixed-precision computing
ResNet
Parallel strategy

ABSTRACT

How to efficiently and quickly train large-scale AI models has become a hot topic in recent deep learning. Mixed-precision training is an effective technique to speed up training and reduce memory usage. At present, the automatic mixed-precision training method mainly uses half-precision (FP16) for the matrix operations of forward and backward propagation of the entire model and accumulates the FP32 weight copies to avoid rounding errors. However, this method is not optimized for each layer individually, leading to poor convergence in large-scale model training because different layers have different data patterns. Therefore, this paper proposes a layered mixed-precision training method, which can flexibly adjust training precisions according to the contribution of each layer to the training effect. Applying the layered mixed-precision method, the ResNet model achieves a 1.9× speedup compared to the baseline and a lower percentage of accuracy loss. In addition, this paper combines the layered mixed-precision method with distributed training strategies. Combining data parallel training, the model achieves a 3.74× speedup by using four Tesla V100 GPUs. The applicability of the layered mixed-precision method in model parallel training has been verified. Combining optimized pipeline parallel training, the model achieves a 3.26× speedup by using three Tesla V100 GPUs.

1. Introduction

Deep learning plays an indispensable role in many fields, including object detection (Redmon et al., 2016), machine translation (Kübler, 2002), and autonomous driving (Grigorescu et al., 2020). Since the introduction of AlexNet (Krizhevsky et al., 2012), the number of layers in neural networks has gradually increased, resulting in a significant increase in the number of parameters. ResNet (He et al., 2016) has become a milestone model in Convolutional Neural Networks (CNNs), with its residual structure allowing for hundreds to thousands of layers in a network. This has led to the development of variants, such as ResNeXt (Xie et al., 2017) and DenseNet (Huang et al., 2017). The residual structure is also common in many large models. However, the increasing complexity of the models leads to higher training costs, including many matrix multiplication operations (Abdelfattah et al., 2020) and parameter updates. Currently, AI models often require large memory and high-performance computing devices, such as graphics processing units (GPUs), for training. When the model is very complex and the size of the training dataset is very large, using a single GPU may take several days to complete training. To achieve faster training, training methods have transitioned from single GPUs to multiple GPUs, including data parallelism, model parallelism (Dean et al., 2012), and pipeline parallelism (Huang et al., 2019) for distributed training.

To improve the training speed, the training method is often changed, which usually reduces the accuracy of model representation and calculation. In deep learning, FP32 is commonly used for training. Currently, automatic mixed-precision (AMP) training (Micikevicius et al., 2017) is the primary training method, which utilizes FP16 to store tensors and perform multiplication and FP32 for accumulation to avoid rounding error. The acceleration of mixed-precision training relies on theunderlying hardware support with Tensor Core architecture. Generally, the speedup of training with a single V100 GPU equipped with Tensor Core architecture can reach about 2× or even higher.

* Corresponding author.
E-mail addresses: lihao2020@cugb.edu.cn (H. Li), wangyz@cugb.edu.cn (Y. Wang), hongyan@email.cugb.edu.cn (Y. Hong), lifei2021@email.cugb.edu.cn (F. Li), xhji@cugb.edu.cn (X. Ji).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.101656
1319-1578/

However, the AMP training method, cannot precisely target the specific precision format for each layer, making it impractical for large-scale model training. In large-scale AI training, models trained with AMP often fail to converge to the loss achieved by FP32 training due to the significant overflow of loss and gradient caused by the heavy utilization of FP16 in mixed-precision training (Ma et al., 2022). Moreover, the adoption of FP16 training can cause a percentage loss in training accuracy. Thus, employing varied precision training formats for individual layers in the network is more suitable for large-scale deep learning training.

Our objective is to propose a novel mixed-precision training approach based on prior research, which is typically employed in training large-scale neural networks. This method involves adapting the computational precision according to the characteristics of diverse convolutional layers in different models. Remarkably, this method speedups model training substantially and reaches rapid convergence.

We propose a layered mixed-precision training method with ResNet as an illustrative example. Specifically, we leverage FP16 training for the residual modules with a high computation load and FP32 optimization training for the fully connected layers that have many parameters. To exploit the potential of each layer for acceleration, convergence, and training accuracy percentage, a layer-by-layer speedup training approach is adopted. Our experimental results show that the proposed layered mixed-precision training can almost match the speedup of AMP training (1.9×) compared to the baseline on a single Tesla V100 GPU. The proposed method also exhibits lower accuracy percentage loss and faster convergence. Furthermore, we extend the layered mixed-precision training to distributed training strategies and combine data parallelism, model parallelism, and pipeline parallelism techniques to achieve superior speedup. The results show that our training method using four Tesla V100 GPUs with data parallelism has a speedup of 3.74× compared to the baseline. The layered mixed-precision training can be well adapted to model parallelism distributed training, and our training method using three Tesla V100 GPUs with pipeline parallelism can achieve an optimization speedup of 3.26×.

The innovations of this paper are as follows:

1) This paper proposes a layered mixed-precision training method for ResNet, which employs distinct optimization levels for residual, convolutional, and fully connected layers. Layered mixed-precision training allows for adjusting the training precision format (FP32, FP16) for each layer separately.

2) The layered mixed-precision training is applied to data parallel distributed training, greatly improving the training speed while effectively controlling the accuracy loss caused by low-precision speedup.

3) The applicability of layered mixed-precision training in model parallelism is also investigated. Pipeline parallelism is employed to optimize model parallel training, thereby further augmenting the training speed for large-scale models.

4) This paper uses multiple benchmark databases for image classification to verify the applicability of layered mixed-precision training method and distributed training strategies in real-life applications. These training strategies have significantly improved the speed of image classification.

The rest of this paper is structured as follows: Section 2 provides a comprehensive review of related research on mixed-precision training. In Section 3, we present a detailed description of the proposed layered mixed-precision training method, along with the ResNet model. Section 4 elucidates the optimization methods for data parallelism, model parallelism, and pipeline parallelism, in combination with layered mixed-precision training. Section 5 presents a discussion of experimental results and performance analysis on speedup. Finally, Section 6 concludes the paper and provides an outlook for future work.

2. Related works

In recent years, the rapid increase in the depth and complexity of AI models has resulted in a correspondingly large rise in computational demands and parameter counts. To alleviate these challenges, numerous acceleration techniques have been introduced, including low-precision computing methods, which are effective in reducing the models' training time and memory consumption. Meanwhile, distributed parallel training strategies are often employed in conjunction with low-precision computing methods to realize further gains in training speed.

Courbariaux et al. (2015) introduced a binary neural network that employs binary values to represent model weights, while FP64 is used to represent the precision of the computation data during training. Subsequently, Courbariaux et al. (2016) extended this method to binarize both weights and activation values while storing other data, such as gradients, in FP32. Hubara et al. (2017) proposed the use of INT2, INT4, and INT6 quantization to train neural networks, which further lowered the training precision. However, these methods are not efficiently applicable to large-scale network models. Gupta et al. (2015) demonstrated the feasibility of training network models on the MINST and Cifar10 datasets using a 16-bit fixed-point representation. Afterward, a joint effort between Baidu and NVIDIA (Micikevicius et al., 2017) resulted in the introduction of the AMP training method, which utilizes FP16 for matrix operations in memory and employs FP32 copies for weight updates, avoiding rounding error, and introducing dynamic scaling techniques to address the issue of gradient underflow.

The development of mixed-precision training techniques continued, with Geng et al. (2020) proposing the Cascaded Mixed-Precision Networks, which combines the binary neural network with mixed-precision networks for joint optimization training. Furthermore, Wang et al. (2018) introduced a block-based stochastic rounding technique for the accumulation of floating-point numbers and successfully achieved 8-bit floating-point training using mixed-precision training, with weights updated in FP16. In addition, researchers from Tsinghua University and Alibaba Damo Academy (Ma et al., 2022), in the BaGuaLu large model training method, observed that AMP cannot be precisely applied to different layers of a neural network. As a solution, they proposed a mixed-precision training strategy based on layers and blocks for the transformer models, which resulted in significant improvements in model convergence and accuracy.

Training models at lower precision require robust hardware support. Jouppi et al. (2017) from NVIDIA Labs introduced the Volta architecture GPU, which utilizes Tensor Core technology to accelerate matrix operations during deep learning model training. The subsequent Volta Tensor Core V100 GPU provides 32 times the throughput of CPUs (NVIDIA, 2023). Using the V100 GPU, training speeds in FP32 outpace those of P100 GPUs by a factor of 2.4. Meanwhile, in FP16, the training effect outperforms P100 GPUs by a factor of 3.7 for ResNet models. Moreover, Choquette et al. (2021) introduced the NVIDIA A100 Tensor Core GPU, which provides superior performance and a larger memory capacity, allowing for even faster training under various scenarios. Specifically, in terms of ResNet models for image classification, the inference speed of A100 GPUs is twice that of V100 GPUs.

In addition, these low-precision or mixed-precision training methods are often combined with distributed parallel training techniques, such as data parallelism, model parallelism, and

pipeline parallelism to further enhance the training speed. The aforementioned research has made a significant contribution to the accelerated training of deep neural network models. In this paper, we propose a layered mixed-precision training method for model training, which is built on the foundation of prior research. By integrating this method with data parallelism, model parallelism, and pipeline parallelism, we introduce a more finely tuned training scheme.

3. Mixed-precision training method

3.1. ResNet model

In the field of deep learning, it is widely recognized that network architectures with greater depth tend to achieve higher training accuracy and better performance for tasks such as image recognition and classification. However, as the network grows deeper, the training difficulty increases, and the problem of vanishing gradients can ultimately lead to decreased prediction performance. To address this issue, He et al. (2016) proposed ResNet, a neural network based on residual learning, which enables the design of deep network architectures with more than 1,000 layers while avoiding the problems associated with network degradation and vanishing gradients.

Residual block refers to a cascade of multiple convolutional layers, each with a parallel connection that accumulates residuals. After the accumulation, the rectified linear unit activation is applied to produce the output of a residual block. In this structure, the concatenation of convolutional layers serves as the residual mapping, while the parallel connection simply bypasses the convolutional layers as identity mapping. The benefit of this design is that it enhances network training performance without adding to the number of model parameters or overall training workload. Additionally, residual blocks can effectively address the problem of network degradation in deep neural networks and lead to optimal training results.

The ResNet network structure comes in five types with varying numbers of layers: 18, 34, 50, 101, and 152. These models typically consist of an input convolutional layer, four major residual blocks, and followed by pooling and fully connected layers for output. Table 1 is the main structure of ResNet18.

3.2. Layered mixed-precision training method

Currently, mixed-precision training methods tend to employ a uniform optimization strategy throughout the model, which may not be optimal. Different layers in a model may have different data patterns and, consequently, may differentially contribute to training performance and model convergence. Therefore, adaptive and layer-specific training methods are necessary for further advances in mixed-precision training. (see Fig. 1).

Table 1
ResNet18 structure.

Layer	Output size	ResNet-18
$Conv1$	112×112	7×7,64,stride = 2 3×3,maxpool,stride = 2
$Conv2\_x$	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
$Conv3\_x$	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
$Conv4\_x$	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
$Conv5\_x$	7×7	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
1×1		average_pool,1000-d fc, softmax

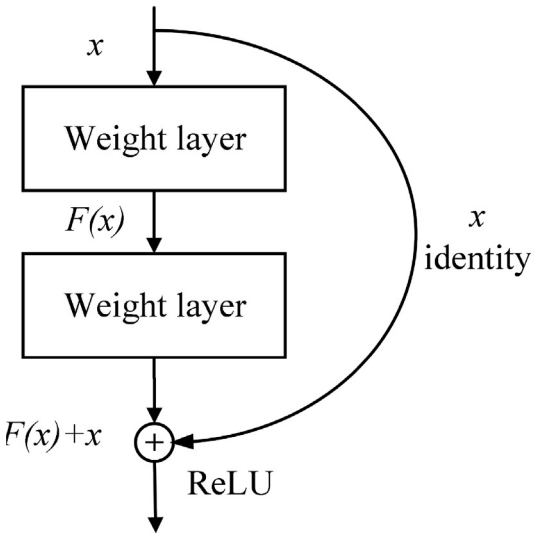


Fig. 1. Residual block.

We propose a novel layer-specific mixed-precision training method. Specifically, we divide all layers of the model into distinct classes, based on their data patterns, and trained each class using precision levels, thereby accelerating the training process and the model convergence. As depicted in Fig. 2, different model components contribute differently to the overall training time across different precision training scenarios (FP16, FP32). For instance, in the ResNet18 structure, a single residual block has a training time that is approximately four times that of the input convolutional layer and output fully connected layer combined. This indicates that

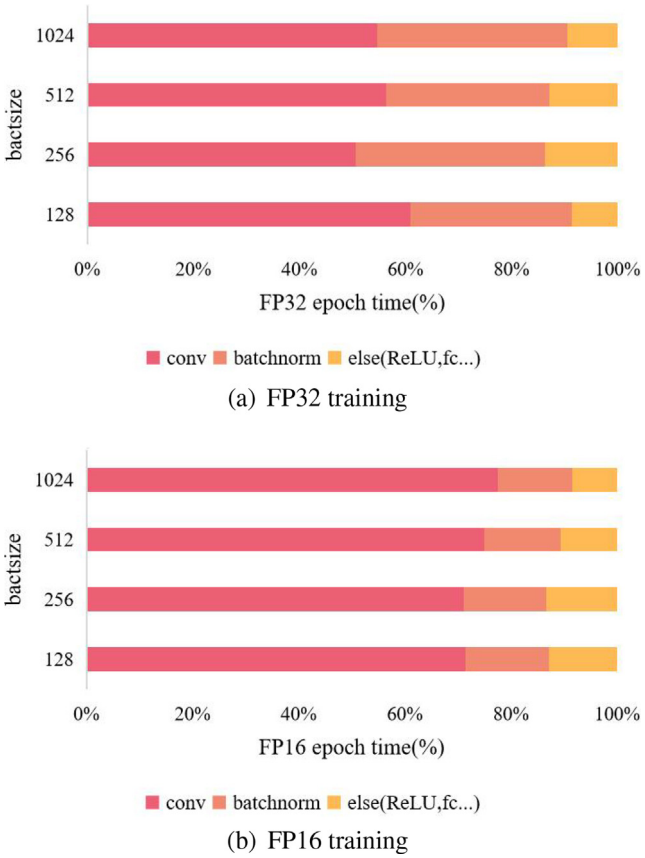


Fig. 2. The contributions of different components to the training time.

the output fully connected layer, which has a significant parameter size relative to the model size, contributes significantly to the ultimate training performance and should not be neglected in optimizing the training.

In this paper, we trained the network using data formats of FP32, FP16. As illustrated in Fig. 3, for the ResNet model, we used FP16 to train the residual layers and FP32 for other layers. Moreover, we continued to employ dynamic loss scaling techniques to address the issue of underflowing caused by small activation function gradients in FP16 training.

4. Parallel training strategy with mixed precision

4.1. Data parallelism

Data parallelism (shown in Fig. 4) is a widely adopted parallel strategy, which is the most common way of training large-scale models. With data parallelism, data is partitioned and distributed to multiple GPUs, each maintaining a local copy of the model parameters. During training, loss function computation and forward propagation are carried out independently on each node. For back propagation, the different processes exchange gradients using a communication mechanism and receive the mean gradients from all the processes. The parameters are then updated by performing gradient descent to complete the training. This method is well supported by most mainstream frameworks.

The foundation of data parallelism is the communication between different GPU processes, which plays an essential role in the parallel training process. There are two common process communication mechanisms: Parameter Server (PS) mechanism (Li et al., 2014) and the Ring All-Reduce mechanism (Gibiansky, 2017). In the PS mechanism, communication occurs around a central node on the GPU, and the communication time increases proportionally to the number of nodes. Therefore, with the PS communication mechanism, training speed for data parallelism decreases inversely with the number of GPU work nodes. In contrast, the communication mechanism of Ring All-Reduce avoids the central node of the GPU. Assuming N GPU processes, gradient updates are divided into two steps (shown in Fig. 5):

1) Scatter-reduce. Each GPU divides the data into N blocks. During each iteration, one data block is sent by each GPU to its right neighbor, and one data block is collected from its left neighbor. It is important to note that the data blocks sent and received by each GPU during each iteration are different. After $N-1$ iterations, the scatter-reduce step is complete, resulting in each GPU having a data block that aggregates the gradients contributed by all GPUs.

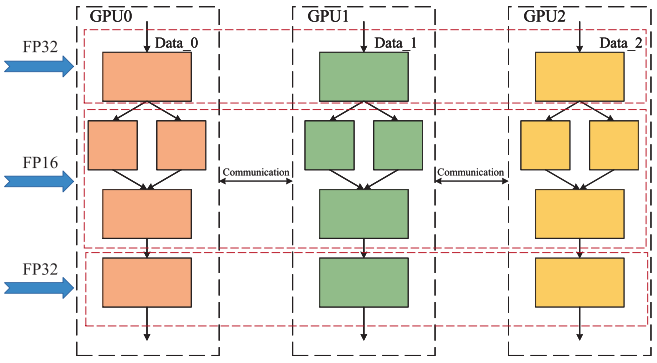


Fig. 4. Data parallelism with layered mixed-precision framework.

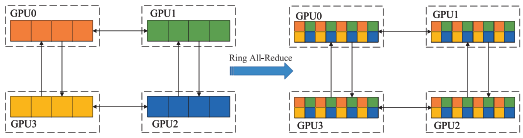


Fig. 5. Ring All-Reduce process.

2) All-gather. In the next $N-1$ iterations, each GPU continues to perform the same data block transmission as in the scatter-reduce step. Eventually, each GPU obtains the gradients for the complete model.

Data parallelism allows the increase of batch size, enabling the training of larger data volumes. This paper optimizes the model training speed by combining layered mixed-precision with data parallel strategies. Specifically, for data parallelism, gradients are transmitted through different layers in the model, and for each layer, mixed-precision training techniques are used for gradient calculations.

4.2. Model parallelism

Model parallel (shown in Fig. 6) is commonly used when a single GPU is insufficient for large models (Coates et al., 2013). Typically, the large model is split by layer and loaded serially onto different GPU work nodes for training. Each GPU node has parameters for different layers. The calculation and update of parameters are divided into multiple GPU nodes in sequence. Therefore, communication between adjacent GPU nodes is required during both

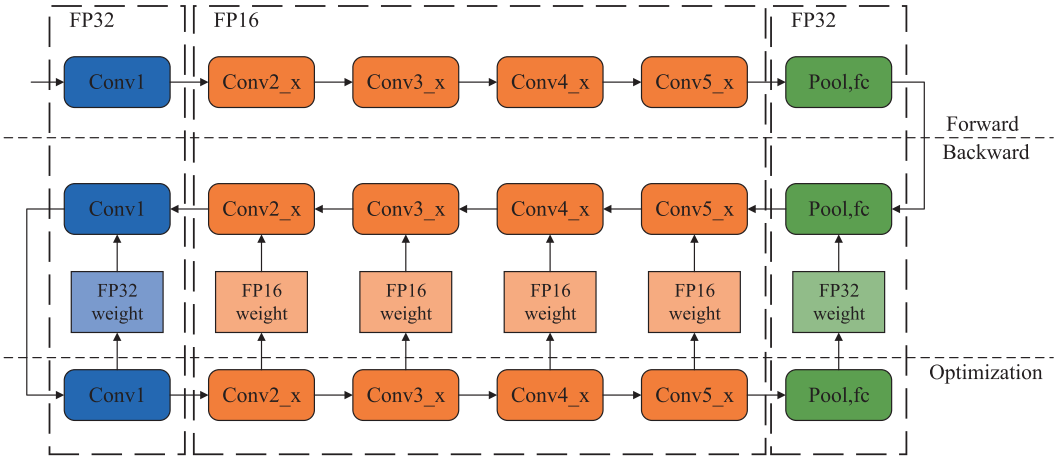


Fig. 3. Layered mixed-precision training for ResNet.

forward and backward propagation to exchange their respective intermediate results. Additionally, another model parallel strategy involves equally distributing all the layer parameters of the network model to different GPU nodes. Since this paper adopts mixed-precision training based on the layer level, this model parallel method is not compatible with mixed-precision training, and the subsequent model parallelism will no longer elaborate on this strategy.

In the convolutional neural network, although the convolutional layers have relatively fewer parameters compared to the fully connected layers, their computational demands are much higher. Additionally, in the ResNet network, the fully connected layers have many parameters. Therefore, to train the ResNet network, a combination of layered mixed-precision training and model parallelism can be adopted. Specifically, the four residual layers can be trained on a single GPU, while the remaining network structure is partitioned and trained across another two GPUs. Furthermore, the training precision (FP32/FP16) can be consistent within each GPU.

In more detail, the calculation of parameters for the *i*-th GPU node requires waiting for the previous node's calculation result in forward propagation and the next node's calculation result in backward propagation. However, frequent computational communication among different GPUs will result in enormous communication and synchronization overheads during training, leading to low GPU utilization rates. This makes the speedup of pure model parallel training far lower than that of data parallelism.

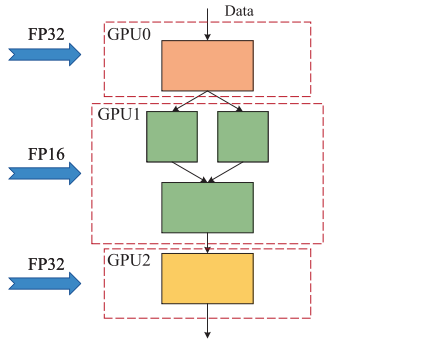


Fig. 6. Model parallelism with layered mixed-precision framework.

Fig. 7. Model parallelism computation approach.

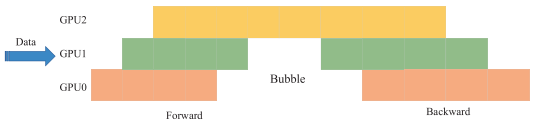


Fig. 8. Pipeline parallelism computation approach.

4.3. Pipeline parallelism

Fig. 7 illustrates that during model parallel training, while one GPU node is performing computation, the other GPUs remain idle. To tackle the issues of low GPU utilization rates and bad speedup caused by communication overhead in model parallelism, the idle time of each GPU during model parallel training can be fully utilized through pipeline parallelism to accelerate the training process.

Pipeline parallelism (Pytorch, 2023) divides the batch size of each training epoch into smaller mini-batches. During the forward pass, each GPU independently performs parameter calculation using mini-batch size data and passes the results to the next GPU, obtaining the loss value in the process. During the backward pass, all GPUs perform gradient descent and parameter updates based on the mini-batch size. The computation process of pipeline parallelism is illustrated in Fig. 8. Compared to model parallelism, utilizing pipeline parallelism can enable simultaneous communication and computation processes. This methodology not only greatly reduces communication overhead but also increases GPU node utilization. To train our ResNet network, we employed a combination of the layered mixed-precision method and pipeline parallelism. It is essential to note that the speed of pipeline parallelism training is dependent on the size of the mini-batch, and therefore, evaluating the speedup of the model under different mini-batch sizes is necessary.

5. Results and discussion

5.1. Experimental setup

This paper conducted experiments of layered mixed-precision method and its parallel strategies on Beijing Paratera Tech@ Supercomputer. We selected the Cifar-10 classification dataset as our experimental dataset and adopted Tesla V100 GPU as the computational device. PyTorch 1.7 was the chosen deep learning framework, with a batch size of 1,024 employed for all model training methodologies. The data parallelism was evaluated by experiments executed on four V100 GPUs. Using SGD descent algorithm and L2 regularization, all models were trained with an epoch set at 100. Experimentation on pipeline parallelism was conducted on mini-batches of 32, 64, 128, 256, and 512, respectively.

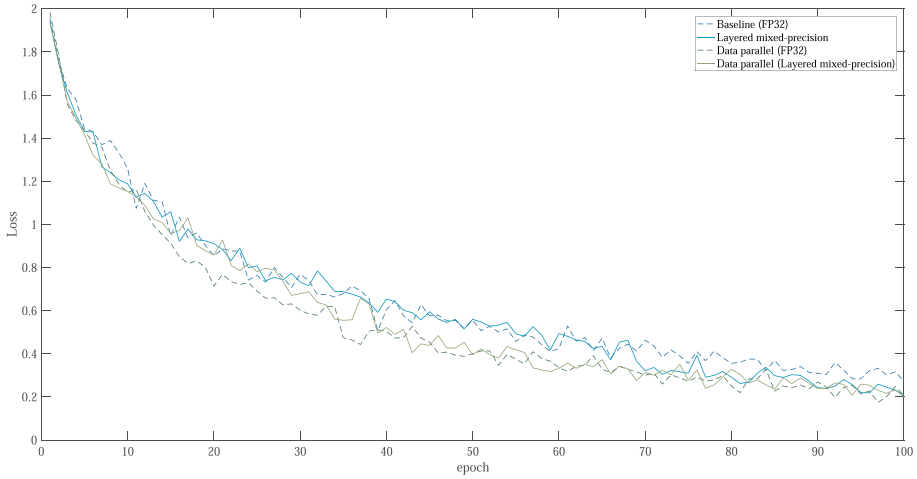
5.2. With layered mixed-precision training

We conducted layered mixed-precision training experiments on the ResNet18 model using a single V100 GPU. To accelerate the residual layers, we adopted a layer-by-layer speedup training approach (as shown in Table 2). Since the input convolution layer, pooling layer, and fully connected layer had significantly more parameters and fewer computational requirements compared to

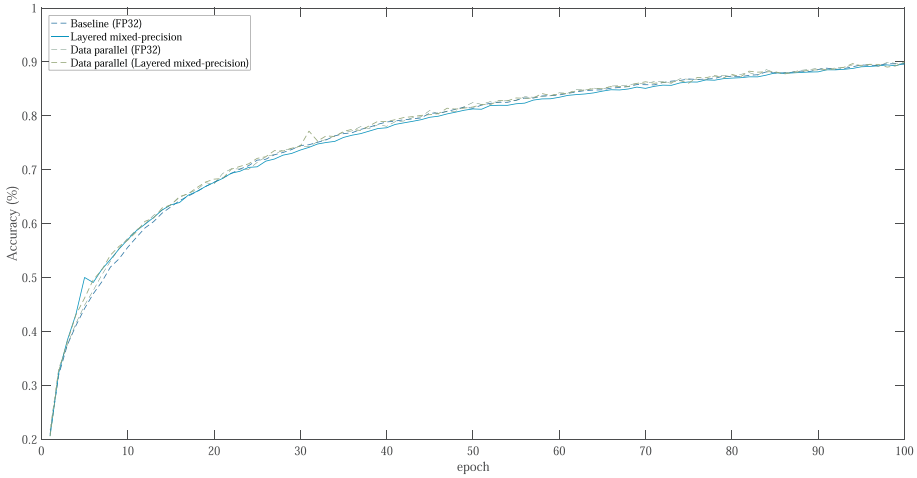
Table 2
Experimental results of layered mixed-precision training.

Layered mixed-precision training	Time (s)	Top-1 accuracy in training set (%)	Top-1 accuracy in testing set (%)	Speedup (×)
FP32 (baseline)	1702.08	89.65	85.07	1.0
1 Resblock for FP16	1441.57	89.60	84.83	1.18
2 Resblock for FP16	1195.20	89.54	84.96	1.42
3 Resblock for FP16	1088.28	89.54	85.17	1.56
4 Resblock for FP16	892.97	89.72	84.77	1.90

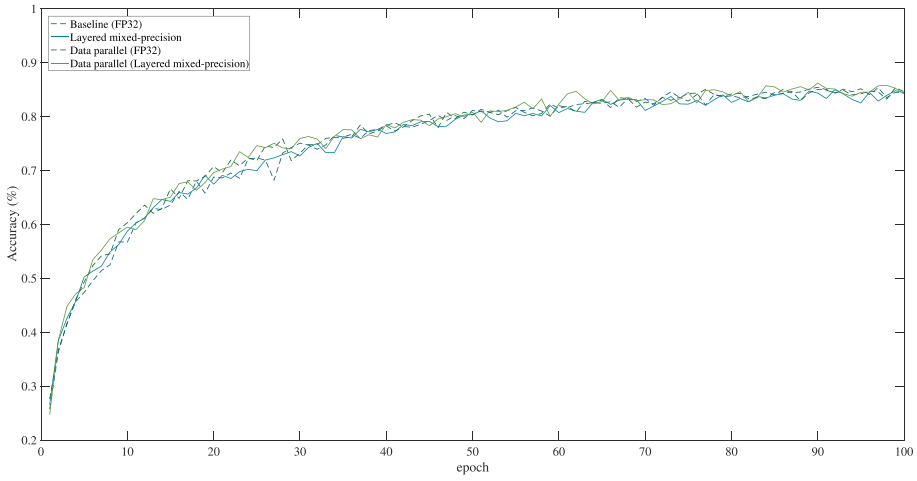
the residual layers, we compensated for the training error by performing FP32 training on these layers. We used the FP32 training model as a baseline and tested the training time, speedup, and top-1 accuracy of both the training and test sets. The experiment's results, presented in Table 2, showed that training with layered mixed-precision for all four residual blocks achieved a 1.9× speedup relative to the baseline. The percentage of accuracy loss in all cases was lower than 1.5%.



(a) Loss



(b) Train accuracy



(c) Test accuracy

Fig. 9. Training results on Cifar10.

Table 3
Experimental results of data parallelism training (4 Tesla V100 GPU).

Data parallel	Time (s)	Top-1 accuracy in training set (%)	Top-1 accuracy in testing set (%)	Speedup (×)
Data parallel (FP32)	576.23	90.37	85.28	2.95
Data parallel (layered mixed-precision)	454.88	89.48	84.72	3.74

Table 4
Experimental results of model parallelism training (3 Tesla V100 GPU).

Model parallel	Time (s)	Top-1 accuracy in training set (%)	Top-1 accuracy in testing set (%)	Speedup (×)
Model parallel (FP32)	1774.00	87.35	81.96	0.96
Model parallel (layered mixed precision)	1427.11	87.10	79.69	1.19

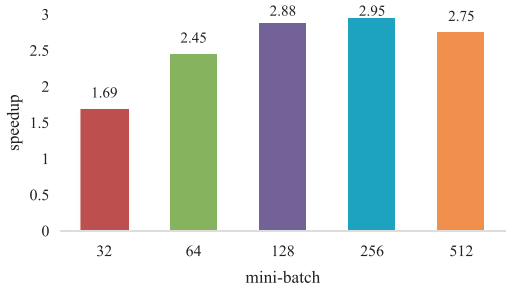


Fig. 10. The speedup performance with different mini-batch sizes.

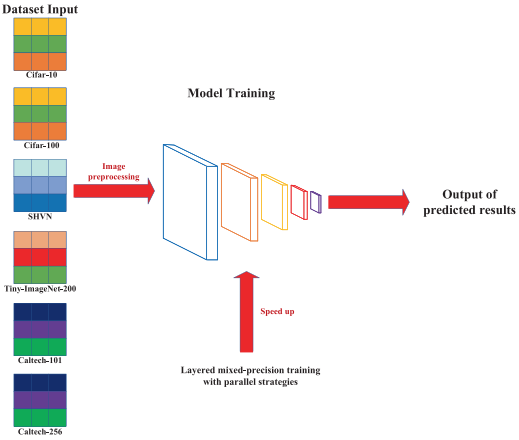


Fig. 11. Application in image classification.

5.3. With parallel training

5.3.1. Results of data parallelism

To achieve better acceleration of model training, we combined layered mixed-precision training with distributed parallel strategies. For the data parallelism strategy, we used four Tesla V100 GPUs, and divided the data non-redundantly into 4×1024 batches for each iteration.

To examine the speedup effect after combining data parallelism, comparative experiments were designed, including FP32 training (baseline), layered mixed-precision training, data parallelism training, and data parallelism combined with layered mixed-precision training. The results of these experiments are shown in Fig. 9 and Table 3.

According to Fig. 9(a), it can be inferred that the convergence speeds of the three experimental sets, in comparison to the baseline, increase successively. The combination of layered mixed-precision training and data parallelism ultimately reaches the smallest loss value. Fig. 9(a), and Fig. 9(c) illustrate the accuracy of the training and testing sets at each epoch, respectively. It is evident that the curves of the four training approaches have extremely similar trends on both sets. The percentage of accuracy loss caused by the model speedup is at a relatively low level. Table 3 presents the specific experimental result. The combination of layered mixed-precision and data parallelism achieves a training speedup of 3.74×, while the percentage of training accuracy loss is lower than 1%. Overall, the results imply that the data parallelism integrated with model training yields significant acceleration effects.

5.3.2. Results of model parallelism

We deployed the layered mixed-precision training method to model parallelism to verify the suitability of their combination. We split the ResNet18 into different parts using three Tesla V100 GPUs and trained it based on the approach presented in Section 4.2. The experimental results are listed in Table 4. The results imply that layered mixed-precision method can be applied to model parallelism. However, due to the low utilization of GPU nodes and communication overhead, the training duration surpassed that of even the baseline. Consequently, we conducted pipeline parallelism experiments to optimize model parallelism in the subsequent experiments.

5.3.3. Results of pipeline parallelism

Regarding the training approach of pipeline parallelism, it is crucial to set an appropriate mini-batch size to enhance model speedup performance. We conducted preliminary experiments using mini-batch sizes of 32, 64, 128, 256, and 512, respectively, and tested the speedup of each group when the epoch was 20. As demonstrated in Fig. 10, the training shows the highest speedup when the mini-batch size is 256. Subsequently, we set the mini-batch size to 256 and conducted experiments using layered mixed-precision training in combination with pipeline parallelism, the results of which are presented in Table 5. The results

Table 5
Experimental results of pipeline parallelism training (3 Tesla V100 GPU).

Pipeline parallel	Time (s)	Top-1 accuracy in training set (%)	Top-1 accuracy in testing set (%)	Speedup (×)
Pipeline parallel (FP32)	716.69	87.03	80.32	2.37
Pipeline parallel (layered mixed-precision)	520.64	87.10	79.89	3.26

Table 6
Experimental results of applications in image classification.

Dataset	Baseline (FP32)	Layered mixed-precision training	Data parallel (4 Tesla V100 GPU)	Pipeline parallel (3 Tesla V100 GPU)
Cifar-100	1.0× 72.10%	2.20× 71.99%	4.02× 70.23%	3.11× 61.42%
SHVN	1.0× 95.36%	1.81× 94.62%	3.21× 95.21%	2.55× 89.94%
Tiny-ImageNet-200	1.0× 65.21%	2.04× 66.01%	3.79× 65.07%	3.35× 56.71%
Caltech-101	1.0× 89.65%	1.91× 89.26%	3.72× 88.76%	2.94× 81.07%
Caltech-256	1.0× 88.54%	1.94× 86.14%	3.67× 87.33%	3.07× 80.21%

successfully exhibit that the combination of layered mixed-precision training and pipeline parallelism can achieve a speedup of 3.26× with high parallel efficiency.

5.4. Application in image classification

In order to verify the efficiency and feasibility of the layered mixed-precision training method in real-life applications, this paper applies this method to image classification. The specific application scenarios of images can include general vision, life scenes, and other aspects. On the basis of the Cifar10 image classification dataset, we have expanded the application of layered mixed-precision methods for image classification. This article introduces more datasets, such as Cifar100, SHVN, Tiny-Image-Net-200, etc. The specific applications are shown in Fig. 11.

These datasets can serve as applications for image classification in different scenarios, and can also verify the feasibility of the layered mixed-precision training method proposed in this paper from more aspects. SHVN, Tiny-ImageNet-200 and other datasets are also supplements to the experimental dataset Cifar10, and can serve as other better benchmark databases for evaluating the training strategies. We tested the speedup and Top-1 accuracy in training set of the above dataset under different training strategies. The experimental results are shown in Table 6.

Table 6 demonstrates that the layered mixed-precision method and parallel strategy can significantly improve the training speed of the model when different datasets are used as benchmark database. It is worth mentioning that the speedup of model training based on layered mixed-precision method on the Cifar-100 reaches 2.20×, combined with data parallelism strategy, the speedup reaches 4.02×.

5.5. Discussion

Previous research lacks exploration on mixed-precision training at the layer level in deep learning. Our proposed method has been experimentally validated to be effective and yield prominent acceleration effects. Compared with the relevant research (Suvojit 0x55aa, 2023), we achieved a speedup of 1.90× for layered mixed-precision training when using a single Tesla V100 GPU, slightly higher than the AMP training, which reached a speedup of 1.88×. Furthermore, when accelerating model training using four Tesla V100 GPUs and combining data parallelism, the layered mixed-precision training was demonstrated to reach a speedup of 3.74×, which was higher than that of AMP training at 3.67×. Moreover, the proposed pipeline parallel training approach contributed to further advancements based on previous research outcome, with a speedup of 3.26×. In summary, the proposed method can be applied in emerging fields, such as real-time model training for face image generation (Pranoto et al., 2022) and embedded machine learning (Ray, 2022).

The layered mixed-precision training method does not include lower data formats, such as FP8 or INT8, which can be used to optimize training in different layers. However, incorporating pipeline parallelism during training can lead to an accuracy loss of around 8%. This might be due to the communication process, suggesting the need for further optimization of the communication mechanism.

6. Conclusion and future work

Deep learning models are evolving towards larger scale networks with increasing model parameters and computations, which can impede the training speed. The traditional AMP training method optimizes the model training solely based on the overall model, leading to a decline in the convergence of training for large-scale models and the persistence of problems, such as many instances of gradient overflow. To tackle these problems, this paper proposes a method called layered mixed-precision training. This method leverages appropriate precision for each layer based on their individual data characteristics and contribution to training performance. For ResNet model applied with this method, residual layers are trained by FP16, while other layers use FP32 for compensation. This method achieves a speedup of 1.9× while maintaining the accuracy of training and test sets. Additionally, we investigate the combination of layered mixed-precision training with distributed training strategies. Specifically, when combined with data parallelism, the speedup obtained with four GPUs is 3.74×, and the convergence speed is faster than that with other previously proposed approaches. We verifie the applicability of layered mixed-precision training to model parallelism by assigning the ResNet model to three GPUs. Furthermore, pipeline parallelism is used to optimize model parallelism and yields a speedup of 3.26× with a mini-batch size of 256.

In terms of future work, there are two main aspects: First, optimizing layered mixed-precision method by introducing lower precision, such as FP8 and INT8, to train models and achieve greater speedup; Second, optimizing communication mechanisms to reduce communication overhead in distributed parallel training.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the China University of Geosciences (Beijing) College Students' Innovation and Entrepreneurship Training Program under Grant 202311415029.

References

Abdelfattah, A., Tomov, S., Dongarra, J., 2020. Matrix multiplication on batches of small matrices in half and half-complex precisions. J. Parallel Distrib. Comput. 145, 188–201.
Choquette, J., Gandhi, W., Giroux, O., Stam, N., Krashinsky, R., 2021. Nvidia a100 tensor core gpu: Performance and innovation. IEEE Micro 41, 29–35.
Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., Andrew, N., 2013. Deep learning with cots hpc systems. In: International Conference on Machine Learning, PMLR, pp. 1337–1345.
Courbariaux, M., Bengio, Y., David, J.-P., 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. Adv. Neural Informat. Process. Syst. 28.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y., 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, arXiv preprint arXiv:1602.02830.
Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al., 2012. Large scale distributed deep networks. Adv. Neural Informat. Process. Syst. 25.
Geng, X., Lin, J., Li, S., 2020. Cascaded mixed-precision networks. In: 2020 IEEE International Conference on Image Processing (ICIP). IEEE, pp. 241–245.
Gibiansky, A., 2017. Bringing hpc techniques to deep learning, Baidu Research, Tech. Rep.
Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G., 2020. A survey of deep learning techniques for autonomous driving. J. Field Robot. 37, 362–386.
Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P., 2015. Deep learning with limited numerical precision. In: International Conference on Machine Learning, PMLR, pp. 1737–1746.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.
Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708.
Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q.V., Wu, Y., et al., 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. Adv. Neural Informat. Process. Syst. 32.
Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y., 2017. Quantized neural networks: Training neural networks with low precision weights and activations, The. J. Machine Learn. Res. 18, 6869–6898.
Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al., 2017. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture, pp. 1–12.
Krizhevsky, A., Sutskever, I., Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Informat. Process. Syst. 25.
Kübler, N., 2002. Teaching commercial mt to translators: Bridging the gap between human and machine. In: Proceedings of the 6th EAMT Workshop: Teaching Machine Translation.
Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., Su, B.-Y., 2014. Scaling distributed machine learning with the parameter server. In: 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14), pp. 583–598.
Ma, Z., He, J., Qiu, J., Cao, H., Wang, Y., Sun, Z., Zheng, L., Wang, H., Tang, S., Zheng, T., et al., 2022. 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, pp. 192–204.
Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G. et al., 2017. Mixed precision training, arXiv preprint arXiv:1710.03740.
NVIDIA, 2023. Nvidia v100 tensor core gpu, https://www.nvidia.cn/data-center/v100/.
Pranoto, H., Heryadi, Y., Warnars, H.L.H.S., Budiharto, W., 2022. Enhanced ipcgan-alexnet model for new face image generating on age target. J. King Saud Univ.-Comput. Informat. Sci. 34, 7236–7246.
Pytorch, 2023. Pipeline parallelism, https://pytorch.org/docs/stable/pipeline.html.
Ray, P.P., 2022. A review on tinyml: State-of-the-art and prospects. J. King Saud Univ.-Comput. Informat. Sci. 34, 1595–1623.
Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.
Suvojit 0x55aa, 2023. Suvojit-0x55aa/mixed-precision-pytorch: Training with fp16 weights in pytorch, https://github.com/suvojit-0x55aa/mixed-precision-pytorch.
Wang, N., Choi, J., Brand, D., Chen, C.-Y., Gopalakrishnan, K., 2018. Training deep neural networks with 8-bit floating point numbers. Adv. Neural Informat. Process. Syst. 31.
Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500.

Hao Li is currently pursuing the B.Eng. degree in electronic information engineering from School of Information Engineering, China University of Geosciences, Beijing. His research interests include high performance computing and parallel algorithm.

Yuzhu Wang received the Ph.D. degree in computer software and theory from University of Chinese Academy of Sciences, in 2015. He is an associate professor at School of Information Engineering, China University of Geosciences, Beijing, China. His research interests include high performance computing and parallel algorithm.

Yan Hong is currently pursuing the M.Eng. degree in computer science from School of Information Engineering, China University of Geosciences, Beijing. Her research interests include high performance computing and parallel algorithm.

Fei Li is currently pursuing the M.Eng. degree in computer science from School of Information Engineering, China University of Geosciences, Beijing. His research interests include high performance computing and parallel algorithm.

Xiaohui Ji received the Ph.D. degree in computer software and theory from Institute of Software, Chinese Academy of Sciences, in 2006. She is an associate professor at School of Information Engineering, China University of Geosciences, Beijing, China. Her research interests include high performance computing and parallel algorithm.