Investigation Related to the Influence of Two Data Parallel Strategies on Pytorch-Based Convolutional Neural Network Model Training

Hao Xue

Software engineering, Hefei University of Technology, Hefei, 230601, China
Email: 2015214433@mail.hfut.edu.cn

Abstract. The escalating prevalence of Convolutional Neural Networks (CNNs) coupled with the incessant growth in both model variants and datasets necessitates the formulation of a judicious data parallelism approach to effectively enhance the pace of model training. This imperative arises as a significant challenge confronted by developers and researchers alike. This paper compares data parallelism and distributed data parallelism. Experiments were designed using the CIFAR-10 and VGG16 models. It is found that the training time of multi-GPU adopting data parallel strategy is not ideal. Analyze the reasons for unsatisfactory training time by studying the impact of hardware and hyperparameters on the data parallel strategy. The data path dependence may be the main reason affecting the training time of the data parallel strategy from the unbalanced GPU usage rate when the data parallel strategy is used. The distributed data parallel strategy training model is compared with the data parallel strategy, and the difference between the results of the two is analyzed. Provides advice on the choice of data parallel strategy. The experimental results show that hardware and hyperparameters are not the main reasons for the unsatisfactory training time of the data parallel strategy. Distributed data parallel strategy training time is better than data parallel strategy, but it needs to ensure the accuracy with multiple GPUs.

Keywords: CNN, Data Parallel, Deep Learning.

1. Introduction

Convolutional neural networks (CNNs) have emerged as a prominent deep learning architecture, particularly for tasks involving computer vision in the last decade [1-4]. It is widely used in transportation, medical care, face recognition, video analysis, natural language processing and many other aspects. In some scenarios, the model has relatively high requirements for training speed, such as automatic driving, Chat GPT language training, and the automatic driving model requires rapid response to road conditions. The GPT-4 model has trillions of parameters. These popular CNNs models require timely feedback to update the model, or because the basic model has a large number of parameters, it is hoped to speed up the training speed of CNNs large models. In recent times, the advancement of storage devices has facilitated the

P. Kar et al. (eds.), *Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2023)*, Advances in Computer Science Research 108,
https://doi.org/10.2991/978-94-6463-300-9_62

frequent updates and expansion of datasets. ChatGPT With the structure of neural network and the framework of constructing neural network structure e.g., TensorFlow and Pytorch determined, how to use limited computer hardware resources to complete a large number of computing tasks has become an important issue. Data parallelism is a common strategy to solve this problem.

PyTorch, an open-source Python library for machine learning, is widely selected as the framework for conducting the research [5]. Deep learning is facing challenges of increasing dataset size and model complexity. Therefore, to train a model effectively and efficiently requires significant computing power. Using a system with multiple GPUs reduces training time, which speeds up application development and further reduces iteration cycles. Teams that can use multiple GPUs to implement data parallelism execution training will have a greater advantage, building models trained on more data in less time, greatly improving the productivity of engineers. PyTorch provides two different data parallelism strategies, Data Parallel (DP) and Distributed Data Parallel (DDP) As CNN model training requires a huge amount of computing. Although data parallelism can address some resource allocation issues, as the number of devices increases, it becomes crucial to investigate whether training results obtained from two different data parallelism strategies remain similar when data is distributed across multiple GPUs. Whether using a data parallel strategy and increasing the number of GPUs will definitely speed up the training processing. Whether multiple GPUs will affect the calculation accuracy due to data parallel processing. Whether the modification of hyperparameters has an impact on the final training results of different data parallelism strategies is a topic worth considering. Although many studies have been done on how to improve the efficiency of the model, they mainly focus on how to optimize the parallel algorithm [6]. Furthermore, it is more biased towards how to optimize the Pytorch DDP strategy [7], and there are fewer studies comparing the training effects of DP and DDP and exploring the reasons.

The purpose of this paper is to study the advantages and disadvantages of different data parallel strategies for CNN training, taking data parallelism in VGG16 of the classic CNN model as an example. It not only explores the influence of different GPU models and different numbers of GPUs on the training speed and accuracy, but also whether the hyperparameters epoch and batch-size designed in the experimental training also affect the experimental results. Then designing an experiment to study whether DDP can improve performance compared to DP. In order to prove the universality, it was also verified on Lenet, which is also a CNN model. This article involves experiments to prove that the use of the DDP strategy is superior to the use of the DP strategy in terms of training speed. Hyperparameters also affect the final experimental results to a certain extent, and finally provide strategy selection suggestions and parameter configuration suggestions for CNN model data parallelism, so that experimenters can take into account training speed and accuracy according to their own needs.

2. Method

2.1 Dataset Description and Preprocessing

The CIFAR-10 data set consists of a collection of 60,000 32×32 color pictures, with a total of 10 categories, distributed across 10 distinct categories. There are 50,000 training images and 10,000 testing images. The dataset is divided into 5 training blocks and 1 testing block, each block has 10,000 images. The test block contains 1, 000 images randomly selected from each category. The training blocks contain random residual images, but some training blocks may contain more for one category than others, and the training blocks contain 5, 000 images from each category.

2.2 Deep Learning Model

This paper selects the VGG16 as a benchmark model [8], which is a deep network model developed by the computer vision group of Oxford University and researchers from Google DeepMind in 2014. The VGG16 network won second place in the classification project of the ILSVRC 2014 competition and the first place in the positioning project. The network's architecture is characterized by its simplicity and strong generalization capabilities when applied to diverse image datasets, making it a popular choice for feature extraction. Additionally, the LeNet-5 is chosen as an experimental test model. LeNet-5 is a convolutional neural network model proposed by Yann LeCun et al. in 1998 [9]. It has significantly influenced the field of deep learning and garnered substantial success in handwritten character recognition tasks. Given that the CIFAR-10 dataset comprises ten distinct classes, adjustments are made to the number of neurons in the final layer of the LeNet-5 model to accommodate the ten output categories.

2.3 Data Parallelism Technology

DP is a data parallelism method proposed earlier, which will open a single process and multiple threads for data parallelization. The training process is divided into forward propagation and back propagation, taking the first forward propagation as an example shown in Fig. 1:

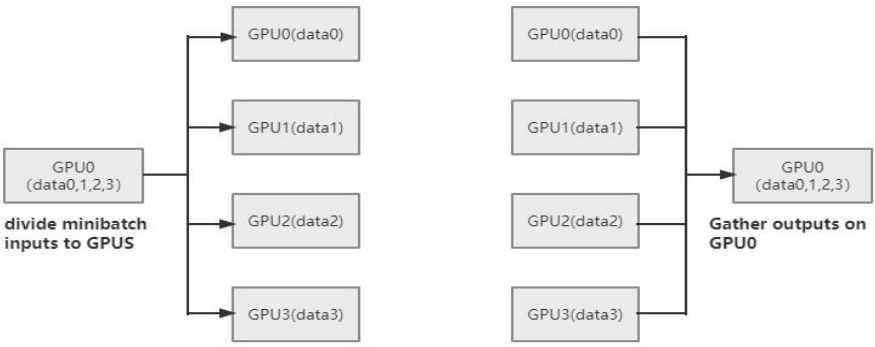

Fig. 1. Forward propagation of DP (Photo/Picture credit: Original).

The model and mini-batch data will be put on GPU:0 (masterGPU). In the first step, GPU:0 will divide the data into sub-mini-batch and scatter to other GPUs. In the second step, GPU:0 will copy its own model parameters to other GPUs. Each GPU has the same model parameters. In the third step, each GPU forward-propagates its sub-mini-batch data on a separate thread to obtain the output of the model. In the fourth step, GPU:0 as masterGPU will collect the output results of all GPUs.

Similar to forward propagation, backpropagation requires GPU: 0 to calculate loss with the real label and get the gradient of loss after getting all the results. GPU: 0 will scatter the loss gradient to all GPUs, and each GPU will calculate the gradient of all parameters according to the backpropagation of the loss gradient. After that, the parameter gradients calculated on all GPUs will be summarized on GPU:0, and GPU:0 will then update the parameters. Thus, completing the model training of a batch. During the whole process, GPU0 only provides data to other GPUs and does not provide labels. After obtaining the calculation results, GPU0 calculates loss and gradients and distributes them to all GPUs to calculate parameter gradients. After GPU0 obtains these data, it updates the parameters.

Compared with DP's single-process multi-thread. DDP strategy, multi-process is selected, there is no MasterGPU, and each GPU performs the same task. Each process loads its own data from disk. Distributed data samplers ensure that loaded data does not overlap across processes. The forward pass and computation of the loss function is performed independently on each GPU. Also, during backpropagation, gradient descent is performed on all GPUs.

2.4 Implementation Details

The optimizer used in the training model in this paper is stochastic gradient descent, with a momentum value of 0.9 and a learning rate of 0.001. On this basis, experiments were carried out on the DP and DDP strategies respectively. Under the DP strategy, the batch-size value was 64, the epoch value was 10, and GPUs selected four 2080ti as the benchmark experiment. Setting different GPU types, batch-size, and epoch values for subsequent experiments. Then the paper uses the DDP strategy to conduct experiments under the same conditions as the benchmark experiment. The conducted experiments analyze and compare the factors contributing to the unsatisfactory training speed observed with the DP strategy, focusing on hardware and hyperparameters. Furthermore, the advantages and disadvantages of the DDP strategy in comparison to the DP strategy are also evaluated and discussed. Subsequent paragraphs, however, are indented.

3. Results and Discussion

3.1 The Performance of the Model Using DP Strategy Based on Multiple GPUs

Adopting the DP strategy to train the VGG16 model with 4 NVIDIA 2080ti graphics cards under the condition that the batch-size, epoch, and other hyperparameters remain unchanged, and the training time and accuracy are shown in Table 1. As the number of GPUs increased, the training time gradually increased, and the accuracy

did not change significantly. Fig. 2 shows the usage of each GPU, indicating GPU0 usage is higher than other GPUs.

Table 1. The influence of number of GPUs in accuracy and time (batch-size=64, epoch=10, GPU:2080Ti, strategy: DP).

Number of GPUs	1	2	3	4
Time(s)	148.73	420.61	890.39	2823.97
Acc(%)	80	79	79	79

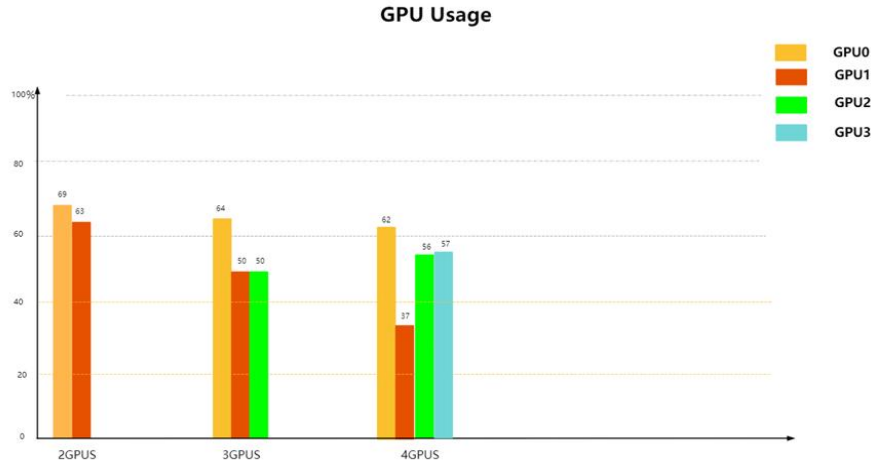

Fig. 2. GPUs usage (Photo/Picture credit: Original).

Multiple GPUs did not produce a significant drop in accuracy compared to a single GPU. However, the purpose of the parallel strategy is to reduce the training time overhead of the model. In this experiment, the time overhead did not decrease but increased in the case of multi-GPUs data parallelism. Obviously, this result does not meet the goal of parallel computing. According to the principle of DP policy, GPU0 distributes data to other GPUs, GPU0 usage is higher than other GPUs.

3.2 The Impact of Hardware on DP Strategy

This paper selects the VGG16 as a benchmark model, which is a deep network model developed by the computer vision group of Oxford University and researchers from Google DeepMind in 2014 [8]. The VGG16 network won the second place in the classification project of the ILSVRC 201

Table 2. The influence of GPU type and number in accuracy and time (batch-size=64, epoch=10, strategy: DP)

GPU	3090×1	3090×2	3090×4	2080Ti×1
Time(s)	93.39	603.27	3464.09	148.73
Acc (%)	79	79	79	80

It can be foparallel,le 2 that the accuracy of model training is still stable without changing the hyperparameters, and the training performance of 3090 on a single GPU is better than that of 2080Ti, but Nvidia-3090 also demonstrated that the more GPUs are used in parallel, the training time increases instead, and even the more powerful the hardware performance, the longer it takes to adopt the same strategy.

Table 2 shows that after changing the GPU to 3090. The greater the number of parallel GPUs, the longer the time overhead. Compared with the previous experiment, the performance of 3090 is worse than that of 2080ti in the case of 2 GPUs and 4 GPUs. Experimental comparison of 3090 and 2080ti under a single GPU proves that the performance of 3090 is stronger than that of 2080ti. Through the comparison of these two experiments, it is found that the reason for the unsatisfactory training time of DP strategy is not the hardware.

3.3 The Impact of Hyperparameters on DP Strategy

Table 3, Table 4 can be compared with the experimental results of Table 1. Increasing the batch-size does not improve the unfavorable situation that the more parallel GPUs, the longer the training time. And as the batch-size increases, the accuracy of model training also decreases.

Table 3. The influence of the number of GPUs on accuracy and time with batch-size=128 (batch-size=128, epoch=10, GPU:2080Ti, strategy: DP)

Number of GPUs	1	2	4
Time(s)	121.52	239.60	1535.89
Acc (%)	77	77	76

Table 4. The influence of the number of GPUs on accuracy and time with batch-size=256 (batch-size=256, epoch=10, GPU:2080Ti, strategy: DP)

Number of GPUs	1	2	4
Time(s)	107,95	149.13	696.53
Acc (%)	74	74	73

Increasing the batch-size can indeed reduce the time spent on training when other conditions remain unchanged. The reason for the suboptimal training time of the DP strategy is not the batch-size. After determined the impact of batch-size on training. This paper next tests the impact of epoch on training.

Table 5. The influence of epoch on accuracy and time with two GPUs (batch-size=64, GPU=2080Ti×2, strategy: DP)

Epoch	3	5	10	15	20
Time(s)	131.35	197.03	420.61	693.42	751.02
Acc (%)	74	78	80	79	80

Table 6. The influence of epoch on accuracy and time with four GPUs (batch-size=64, GPU=2080Ti×4, strategy: DP)

Epoch	3	5	10	15	20
Time(s)	766.25	1281.60	2823.97	4727.20	5549.49
Acc (%)	74	78	79	79	80

According to Table 5, and 6, with the increase of epoch, although the accuracy of model training increases, the model training accuracy tends to converge after reaching a certain Epoch value, while the time spent training the model still increases almost linearly.

Similar to Table 3 and 4, adjusting the epoch value does not solve the problem that the DP strategy makes the time spent of multiple GPUs longer than the time spent of a few GPUs.

3.4 The Performance of the Model Using DDP Strategy Based on Multiple Gpus

After excluding hardware and hyperparameter factors, this paper considers whether the DP strategy itself has defects. The following design experiment adopts the DDP strategy for VGG16. It can be seen from Table 7 that the time spent on training the VGG model using the DDP strategy in Table 7 is more in line with the actual purpose of data parallelism: reducing the time spent on model training. However, different from the DP strategy, the model trained by the DDP strategy begins to decrease as the number of GPUs increases. previous Tables has confirmed that as the batch-size increases, the speed can be improved, and the model accuracy can be reduced. As the epoch increases, while the training time increases linearly, the accuracy of the model increases with time and finally converges. To confirm that the DDP strategy does reduce the accuracy of the parallel model under multiple GPUs, Table 8 is designed again: replace the LeNet-5, which is also a CNN model, for verification.

Table 7. The influence of number of GPUs in accuracy and time with VGG16 (batch-size=64, epoch=10, GPU:2080Ti, strategy: DDP, Model: VGG16)

Number of GPUs	1	2	3	4
Time(s)	148.30	90.52	65.31	56.15
Acc(%)	79	78	76	73

Table 8. The influence of number of GPUs in accuracy and time with LeNet-5 (batch-size=64, epoch=10, GPU:2080Ti, strategy: DDP, Model: LeNet-5)

Number of GPUs	1	2	3	4
Time(s)	116.15	67.23	48.75	44.21
Acc (%)	52	51	48	44

The similar results obtained in Table7 and Table 8 further indicate that the DDP strategy is very effective in shortening the training time of the model, but it still faces the problem of multi-GPU parallelism to improve performance but reduce accuracy.

3.5 Discussion on the Reasons Affecting the Performance of DP and DDP

The DP strategy may have many adverse effects from the perspective of saving time and cost, even from the exploration of hyperparameters such as batch-size and epoch, it is still difficult to make the DP strategy reduce the model training time. The DDP model has achieved the task of reducing training time very well. For the DP strategy, it does not reduce the time overhead but increases it. This article is guessed as follows:

Data dependence: DP only has MasterGPU to load data from the disk [10], and the MasterGPU must distribute the summary data every time it is forwarded and backward propagated. time. The remaining GPUs are data dependent on the MasterGPU, and their computing speed is greatly limited by the Master GPU in the DP strategy. Fig. 2 also shows that the utilization rate of the MasterGPU is significantly higher than that of other GPUs.

The time overhead weighted by communication cost is too large: unnecessary times of scatter and gather data is a huge time overhead. There is no such huge time overhead in the case of DDP strategy and single card.

Similarly, while DDP reduces the time overhead, the accuracy of model training in multi-GPU scenarios is also reduced. It is possible that the DDP strategy evenly distributes a batch of data to multiple GPUs, and independently calculates the loss and gradient on each GPU. If it is not synchronized BN, then the mean and variance calculated on each GPU are only for the data allocated by this GPU, which will lead to inconsistent output of the BN layer and affect the model output accuracy.

4. Conclusion

In this paper, the effects of two data parallelism strategies DP and DDP on the model training performance under the Pytorch architecture are verified through experiments. Designed experiments to study the reasons that affect the performance of DP strategy from the aspects of hardware and hyperparameters. Analyzed the reason why the training time of DP strategy is not ideal in multi-GPU parallelism. The experimental results show that the unsatisfactory training time of DP strategy has little correlation with hardware and hyperparameters. Compared with the DP strategy, the DDP strategy can significantly improve the training speed when using multiple GPUs, but it needs to pay attention to maintaining the training accuracy. In the future, further study plans to test more hyperparameters and data sets to explore when the DP strategy can effectively reduce the training time and when it will increase the time overhead, so as to make recommendations when implementing data parallelism.

References

1. Kayalibay, B., Jensen, G., et al.: CNN-based segmentation of medical imaging data. arXiv preprint arXiv:1701.03056 (2017).
2. Qiu, Y., Wang, J., Jin, Z., et al.: Pose-guided matching based on deep learning for assessing quality of action on rehabilitation training. Biomedical Signal Processing and Control, 72: 103323 (2022).
3. Nguyen, T., Nguyen, G., Nguyen, B. M.: EO-CNN: an enhanced CNN model trained by equilibrium optimization for traffic transportation prediction. Procedia Computer Science, 176: 800-809 (2020).
4. Jiang, H., et al: Face detection with the faster R-CNN. 2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017). IEEE, 650-657 (2017).
5. Paszke, A., Gross, S., Massa, F., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32 (2019).
6. Tang, Z., Shi, S., Chu, X., et al.: Communication-efficient distributed deep learning: A comprehensive survey. arXiv preprint arXiv:2003.06307 (2020).
7. Li, S., Zhao, Y., Varma, R., et al.: Pytorch distributed: Experiences on accelerating data parallel training. arXiv preprint arXiv:2006.15704 (2020).
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
9. Le, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324 (1998).
10. Li, S., Zhao, Y., Varma, R., et al.: Pytorch distributed: Experiences on accelerating data parallel training. arXiv preprint arXiv:2006.15704 (2020).