FreezePipe: An Efficient Dynamic Pipeline Parallel Approach Based on Freezing Mechanism for Distributed DNN Training

Caishan Weng[1], Zhiyang Shu[2], Zhengjia Xu[1], Jinghui Zhang[1], Junzhou Luo[1], Fang Dong[1], Peng Wang[3], Zhengang Wang[4]

[1]School of Computer Science and Engineering, Southeast University, Nanjing, P.R. China, [2]Boston College
[3]Data Science Center of Excellence, Deloitte Consulting(Shanghai) Co., Ltd, Shanghai China,
[4]Jiangsu HHigh-Tech Software Technology Co.Ltd

Outline

1 Background

2 FreezePipe

3 Implementations

4 Conclusion

Background: AI impacts our life

Background: Learning in distributed manner

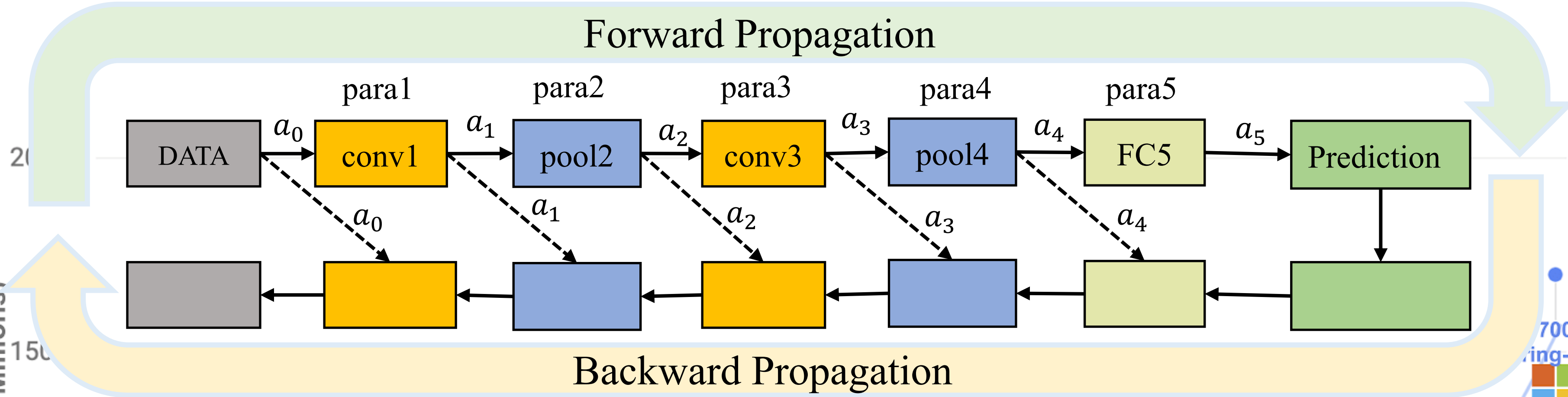
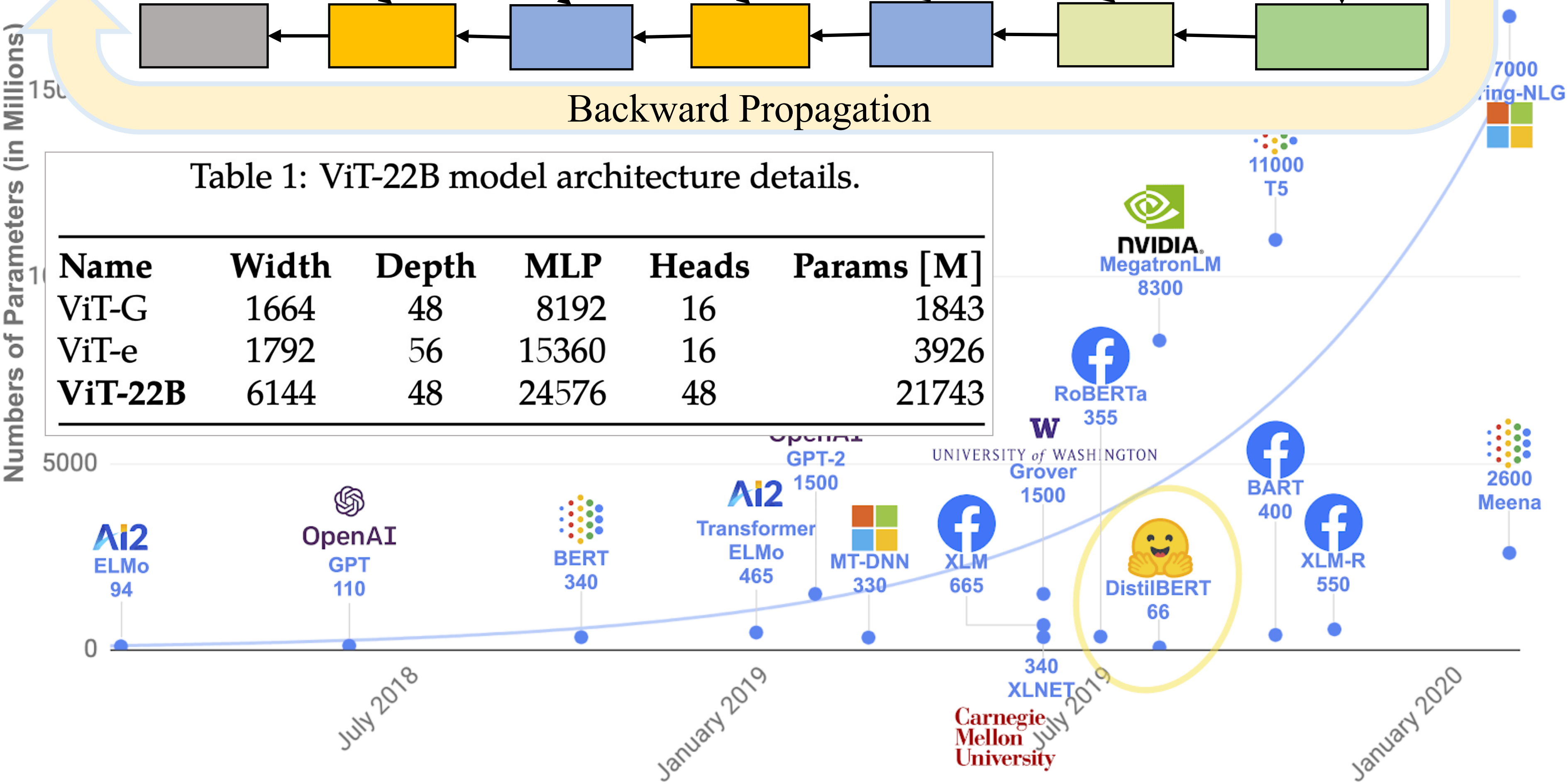
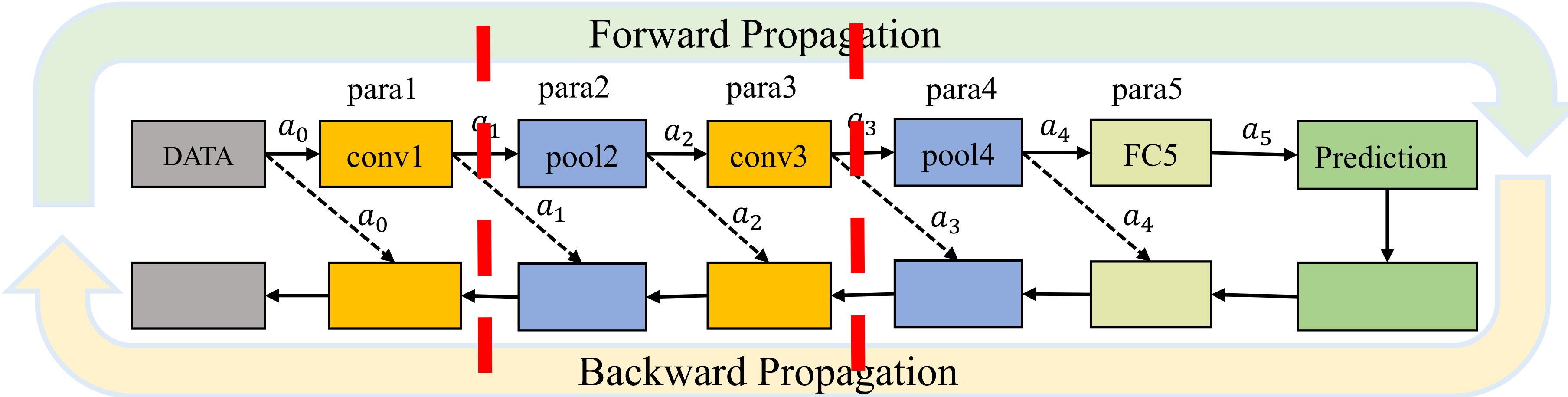


Table 1: ViT-22B model architecture details.

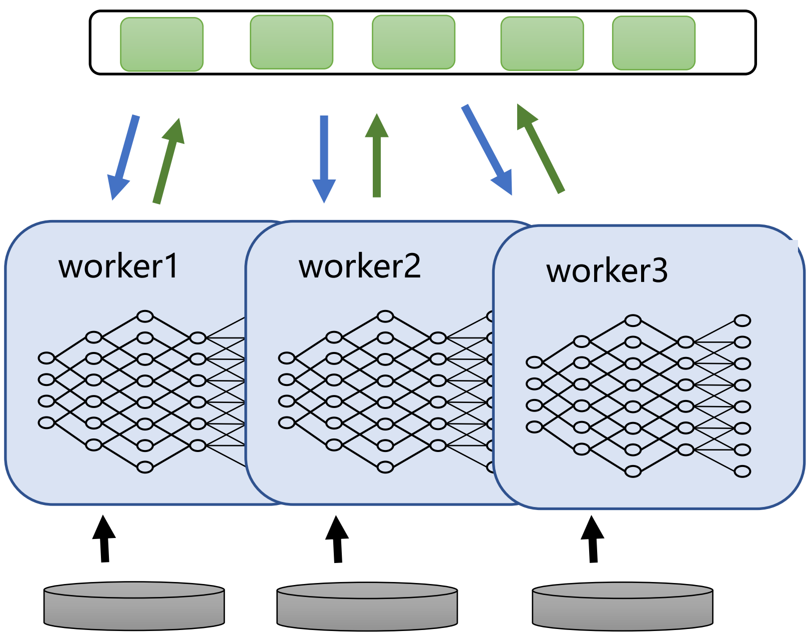
Name	Width	Depth	MLP	Heads	Params [M]
ViT-G	1664	48	8192	16	1843
ViT-e	1792	56	15360	16	3926
ViT-22B	6144	48	24576	48	21743



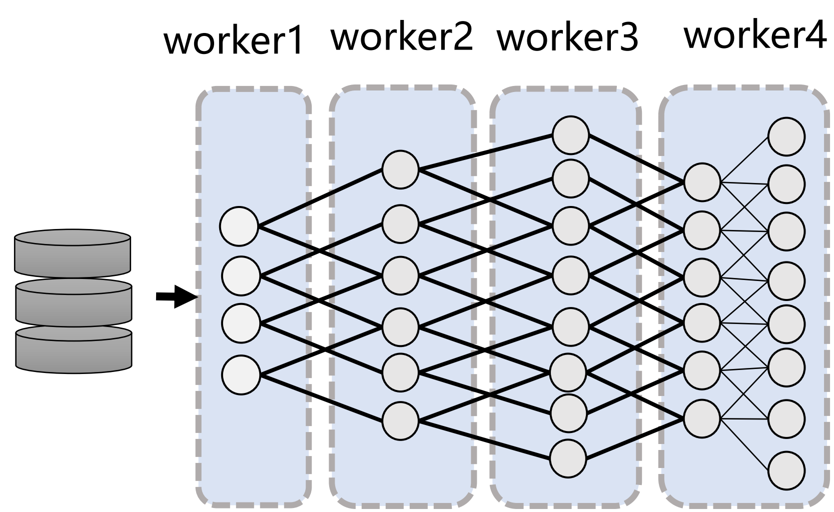
Background: Learning in distributed manner



1.Data Parallelism

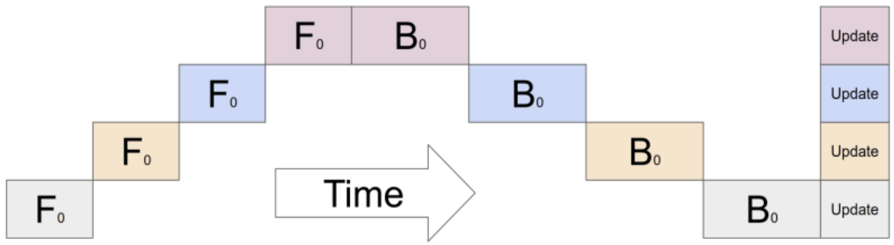


2.Model Parallelism



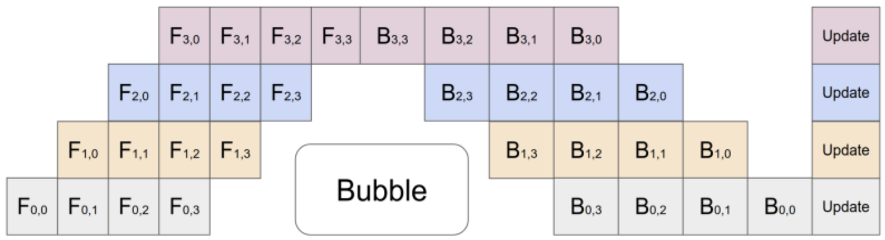
Background: Bubble time can be reduced

Model Parallelism



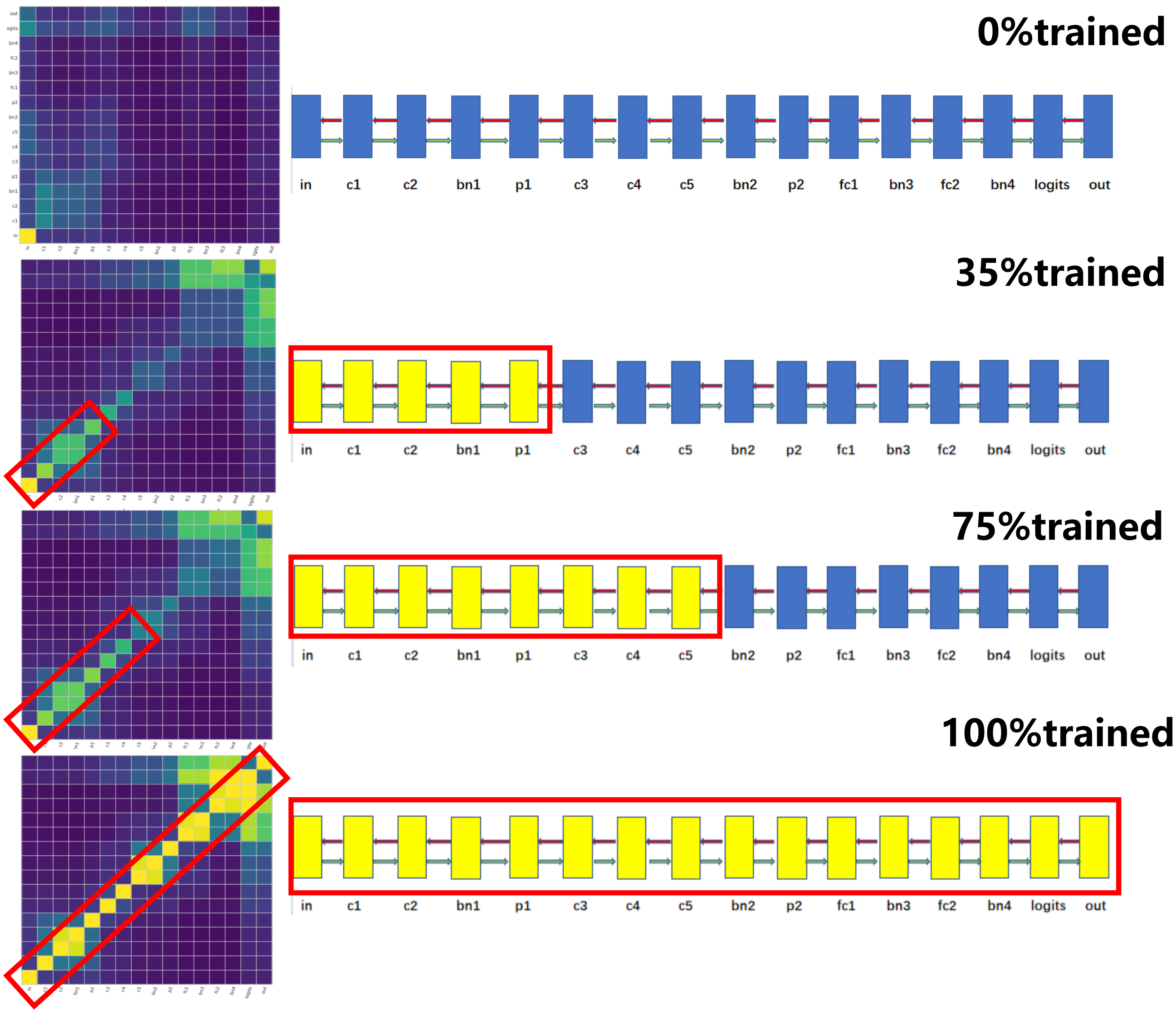
➢ A low GPU utilization due to the neural network's computational dependency, which adversely impacts the efficiency of distributed training.

Pipeline Parallelism



➢ Using pipeline parallelism, each batch of data is further split into several micro-batches, and each micro-batch is processed by a pipeline composed of different parts of the model placed among the GPU cluster, so-called a stage, reducing GPU idle time.

Background: How the models converge?

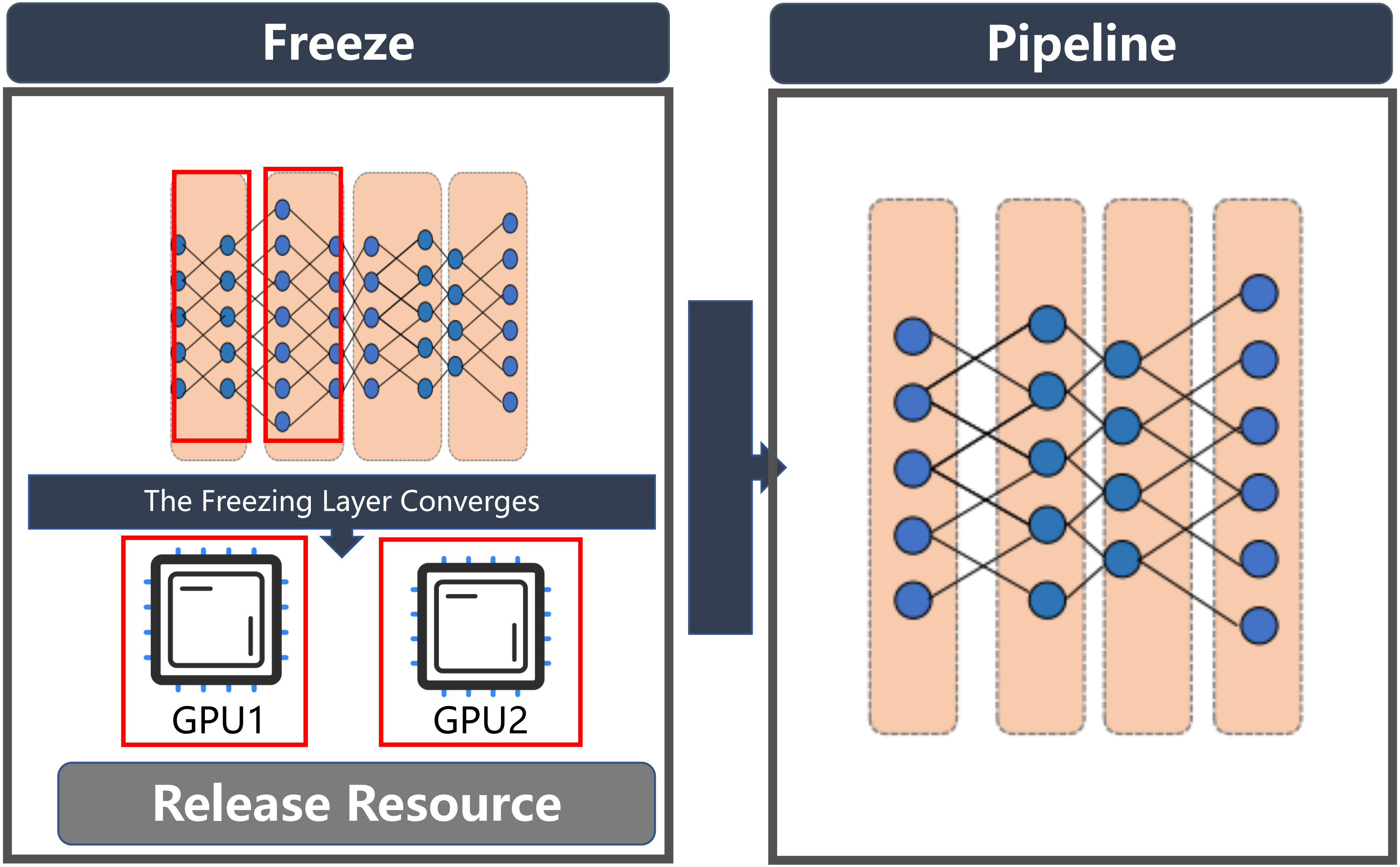


- **Why can be frozen?**

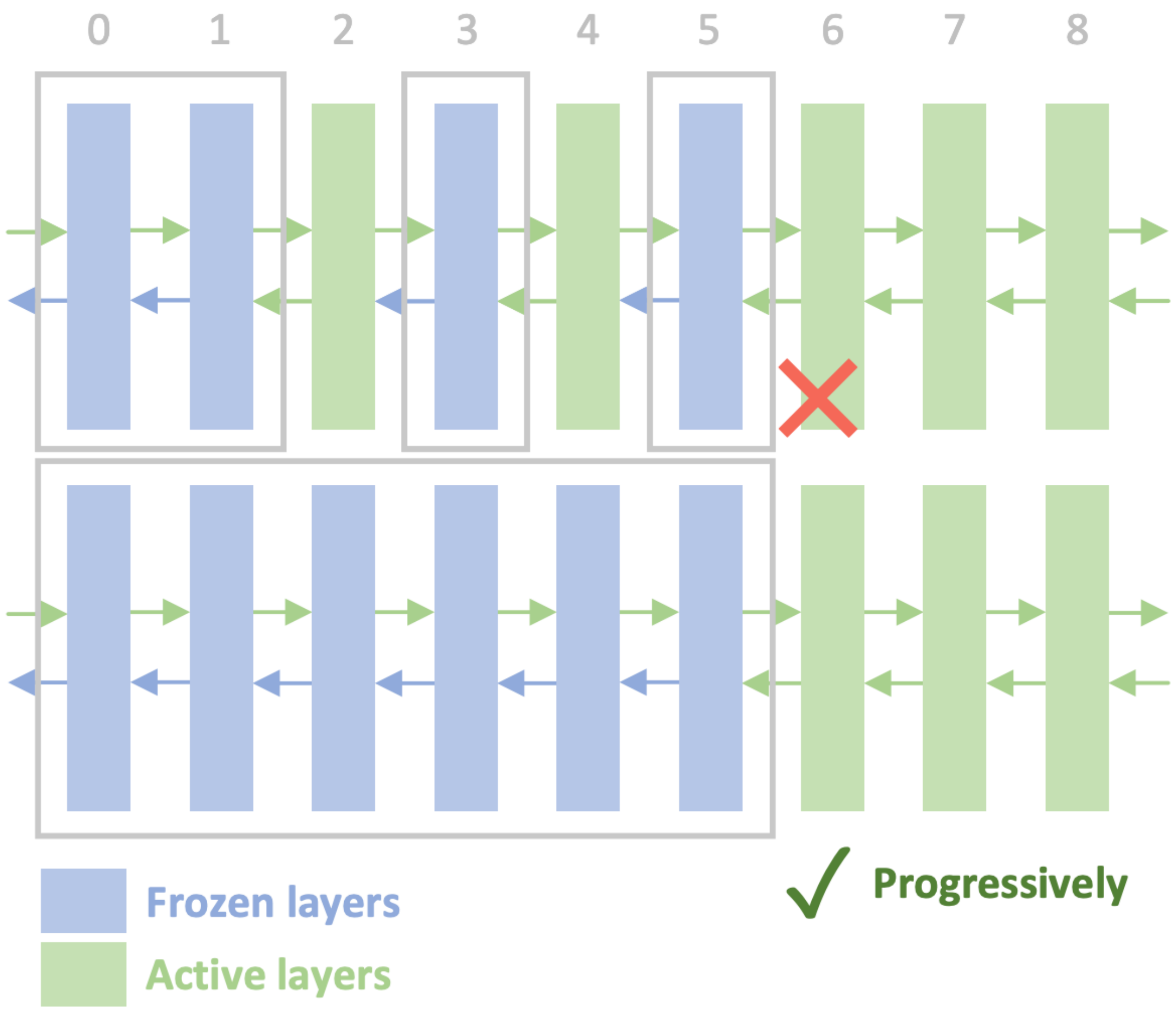
As the number of training iterations increases, the neural network ,gradually converges from the input layer to the output layer.

[1] Raghu M, Gilmer J, Yosinski J, et al. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017: 6078-6087.

FreezePipe: Overview



FreezePipe: Freeze Decision



FreezePipe: Freeze Decision

Freezing rate:

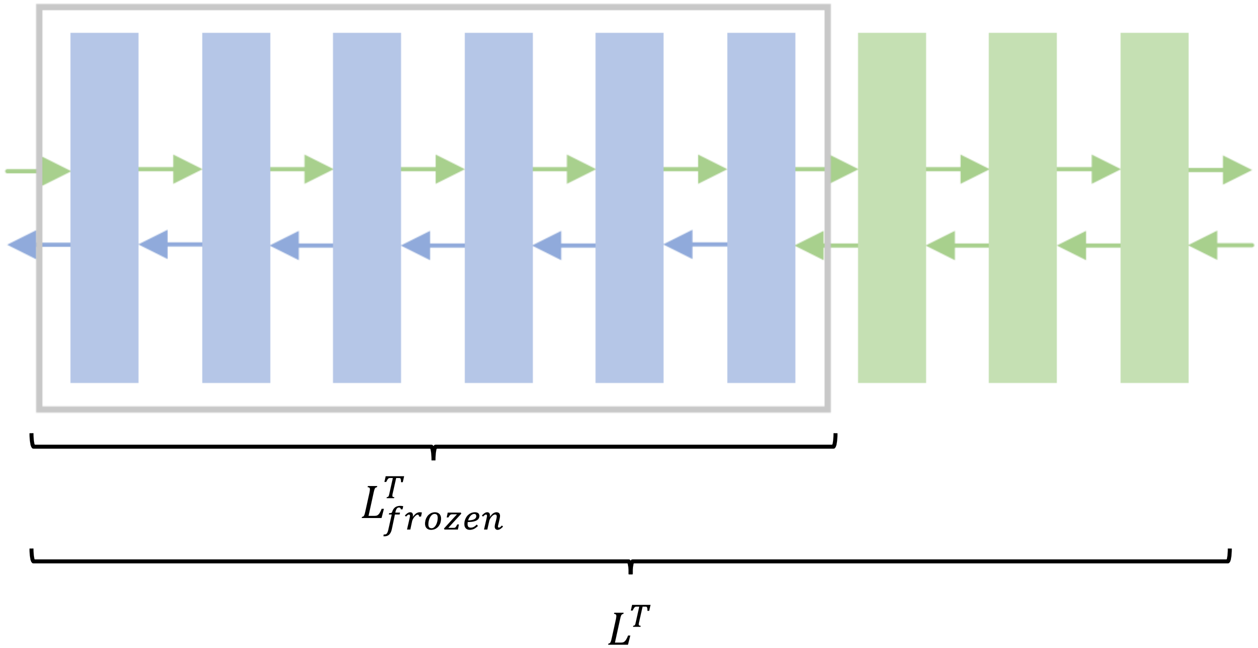
$$\beta^T = \frac{L_{frozen}^T}{L^T}$$

Cumulative gradient difference rate:

$$\eta_l = \frac{\left\| \sum_{i=1}^{m} G_{l_i} \right\|}{\left\| \sum_{i=1}^{m} |G_{l_i}| \right\|}$$

Freezing decision:

$$\eta_i \le \eta_{\beta^T L^T}$$



FreezePipe: Freeze Decision

Freezing rate:

$$\beta^T = \frac{L^T_{frozen}}{L^T}$$

Cumulative gradient difference rate:

$$\eta_l = \frac{\left\|\sum_{i=1}^{m} G_{l_i}\right\|}{\left\|\sum_{i=1}^{m} |G_{l_i}|\right\|}$$

Freezing decision:

$$\eta_i \leq \eta_{\beta^T L^T}$$

Algorithm 1 Online Freezing

Require: Gradient difference list $\{\eta_1, \eta_2, \ldots, \eta_L\}$,percentile β^T
Ensure: Number of frozen layers l

```
for i = 1 to L do
    if η_i <= η_{β^T} then
        freeze layer_i
    else
        l = i − 1,break
    end if
end for
```

Lightweight Freeze Determination System:

We employ distributed gradient accumulators within each GPU to aggregate inter-layer gradients, mitigating the impact of extensive gradient computations on GPU utilization. Subsequently, by calculating the gradient norms on the CPU, we determine the frozen layers

Dynamic Training Module: Model Partition

The computation time for forward propagation and backward propagation of the *i*-th layer partition:

$$comp_i = \sum (compF_i + compB_i)$$

Assuming *N* GPUs, *M* micro-batches, the total training time *A* for GPipe is:

$$A \approx (M + N - 1) * \max\{comp_i\} + T_{comm} + T_{update}$$

The GPipe schedule is dependent upon the maximum value of computing time at all stages:

$$\max\{comp_i\} \rightarrow A$$

Thus, the problem is transformed into a MiniMax problem:

$$\min \max\{comp_i\} \rightarrow \min A$$

Dynamic Training Module: Model Partition

Algorithm 2 Integer Binary Search

Require: time costs list $sequence = \{\mu_1, \mu_2, \ldots, \mu_L\}$,the number of stages $partitions$

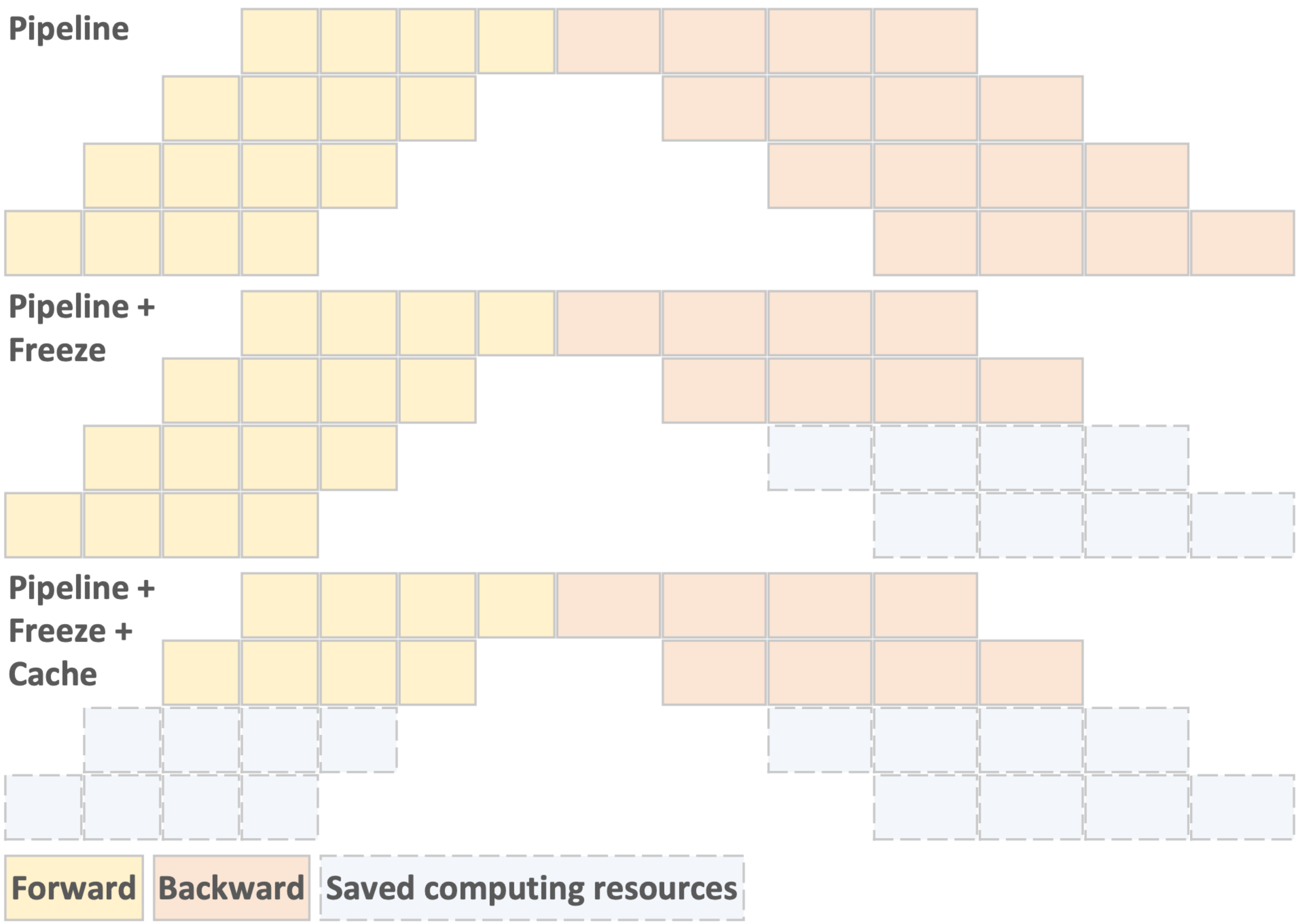
Ensure: partitionslist

```
for i = 1 to L do
    μ_i = [μ_i * 1000]
end for
low = max(sequence)
high = sum(sequence)
while low < high do
    mid = low + (high − low)//2
    nowpartitions = getpartition(sequence, mid)
    if nowpartitions <= partitions then
        high = mid
    else
        low = mid + 1
    end if
end while
maxvalue = low
for i = 1 to Len(sequence) do
    update partitionslist
end for
```

2) *Optimal Solution:* An integer binary search method is designed to solve the Min-Max problem. Input to the algorithm is a list of time costs for each layer of the neural network and the number of stages in the pipeline, as follows: This algorithm has a time complexity of $O\left(N \log\left(\sum_i^n \mu_i\right)\right)$, which is less than the conventional dynamic programming algorithm $O(kn^2)$ and the Torchgpipe [12] algorithm "Block Partitions of Sequences" $O(kn^3)$.

Dynamic Training Module: Forward Cache

Automatic Caching：



Repartition after caching

After caching the intermediate activations of the frozen layers, as a result of the freed GPU computational resources, the pipeline needs to be repartitioned :

$$A = \begin{cases} A^{AfterCaching}, max\{\mathrm{comp}_i^G\} > max\{\mathrm{comp}_i^{AfterCaching}\} \\ A^G, otherwise \end{cases}$$

Implementation

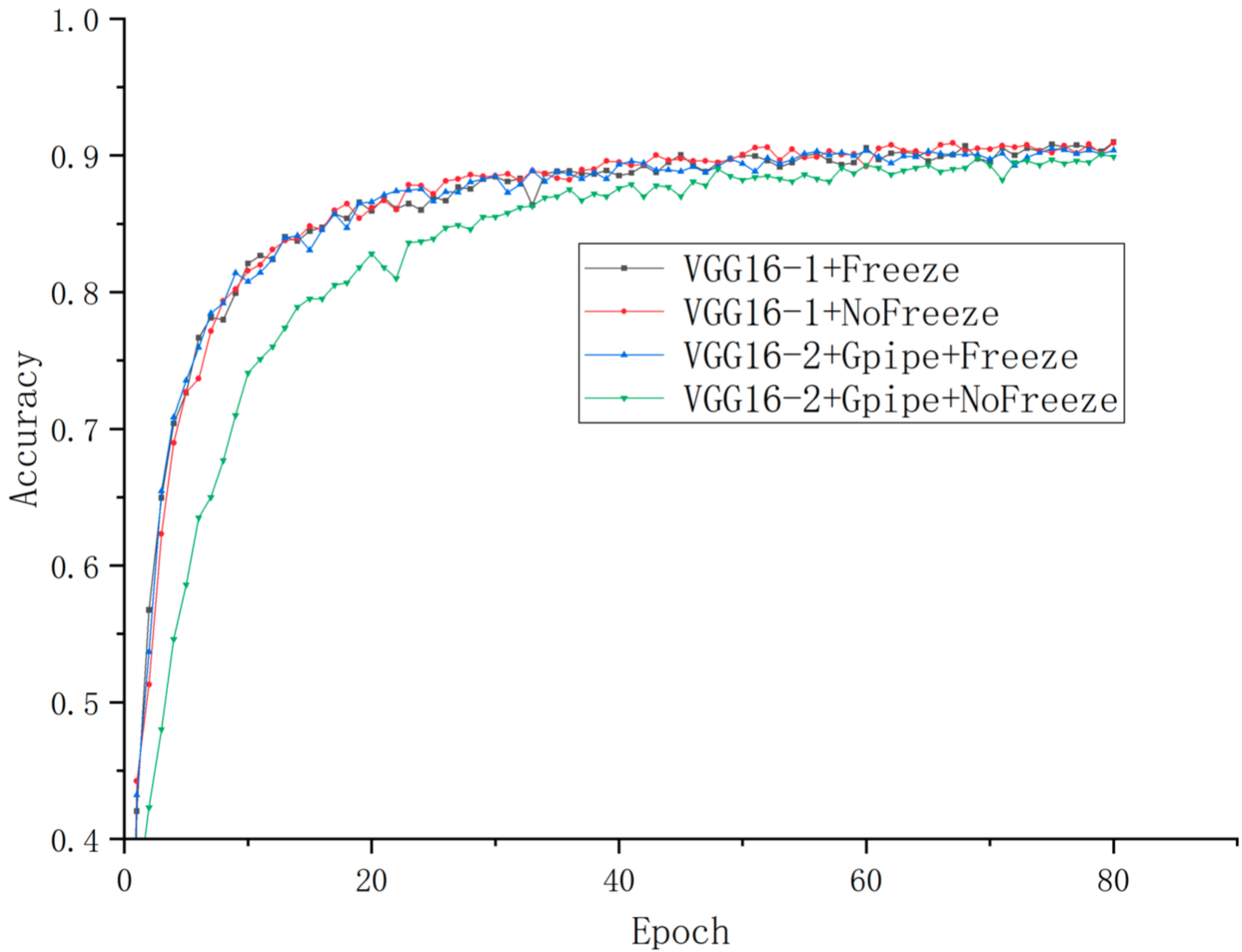
Benchmarks:
Our primary performance metrics focus on the training time and accuracy of a deep neural network (DNN) model. In this section, we compare FreezePipe to TorchGPipe[1] through experiments. We utilize the VGG-16 architecture with the CIFAR-10 dataset. A batch size of 64 is employed for each DNN model.

Hardware configuration:
Our primary experiments utilize a GPU cluster comprising individual machines, each outfitted with an NVIDIA GeForce RTX 3090, boasting 24GB of GPU memory. The inter-GPU bandwidth within a machine, utilizing the PCIE protocol, is 10GB/s. All machines operate on the 64-bit Ubuntu 20.04 operating system, equipped with NVIDIA-SMI 470.141.03, Driver Version 470.141.03, and CUDA Version 11.4.

Hyperparameters: In the experiment, we fixed the freezing rate to 0.5 and performed online freezing every ten epochs, ideal results yielded.

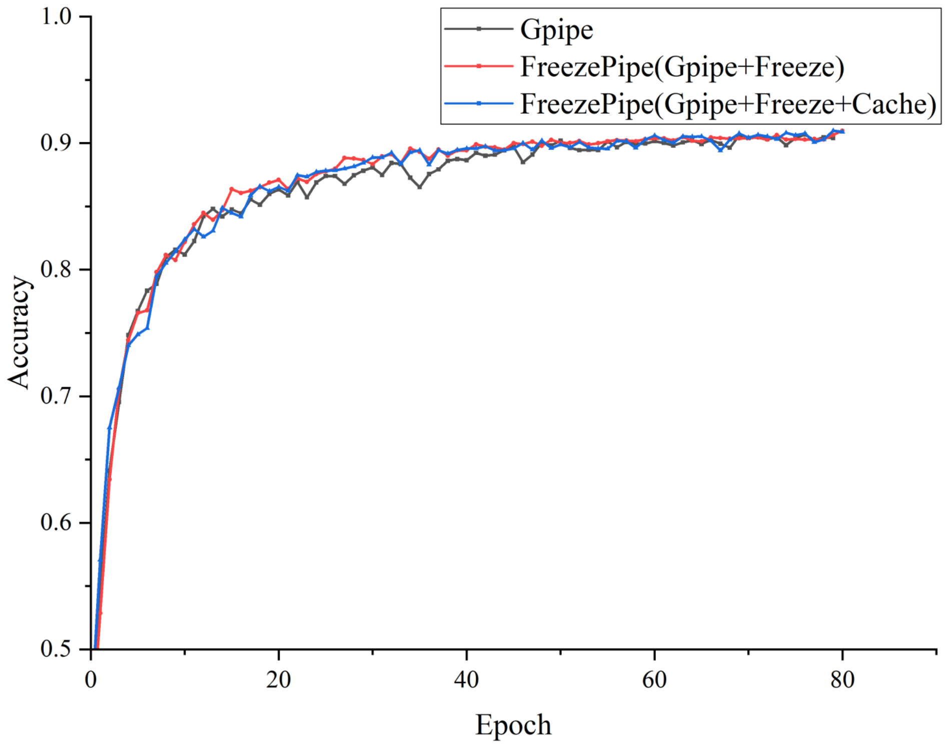
[1] Kim C, Lee H, Jeong M, et al. torchgpipe: On-the-fly Pipeline Parallelism for Training Giant Models[J]. arXiv e-prints, 2020: arXiv: 2004.09910.

Conclusion

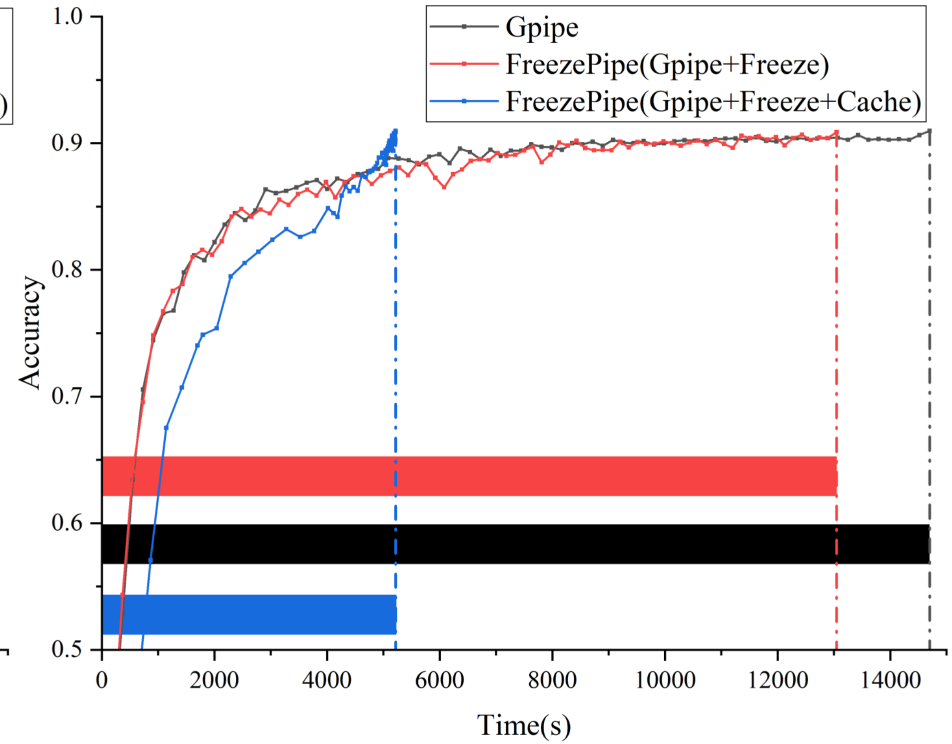


Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
Epoch
VGG16-1+Freeze
VGG16-1+NoFreeze
VGG16-2+Gpipe+Freeze
VGG16-2+Gpipe+NoFreeze

Conclusion



Gpipe
FreezePipe(Gpipe+Freeze)
FreezePipe(Gpipe+Freeze+Cache)
Accuracy
Epoch



Gpipe
FreezePipe(Gpipe+Freeze)
FreezePipe(Gpipe+Freeze+Cache)
Accuracy
Time(s)

Thank You For Your Listening

東南大學
SOUTHEAST UNIVERSITY