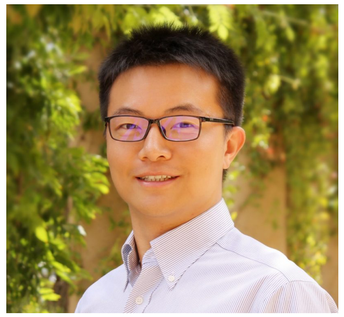
MIT HAN LAB

EfficientML.ai Lecture 20 Distributed Training

Part II



Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA

@SongHan_MIT

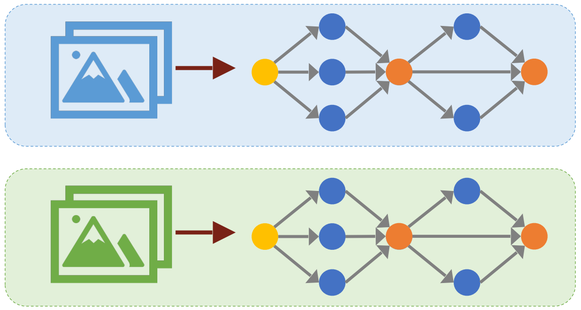
Lecture Plan

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. Understand the bandwidth and latency bottleneck of distributed training
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. Delayed gradient update: overcome the latency bottleneck

Lecture Plan

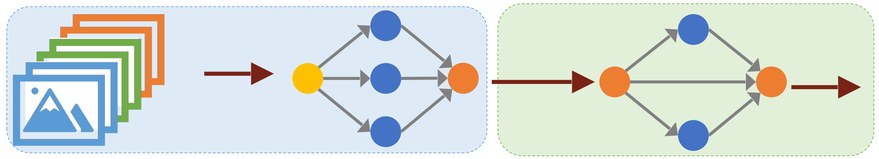
1. **Hybrid (mixed) parallelism and how to auto-parallelize**
2. Understand the bandwidth and latency bottleneck of distributed training
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. Delayed gradient update: overcome the latency bottleneck

Different Parallelization Methods



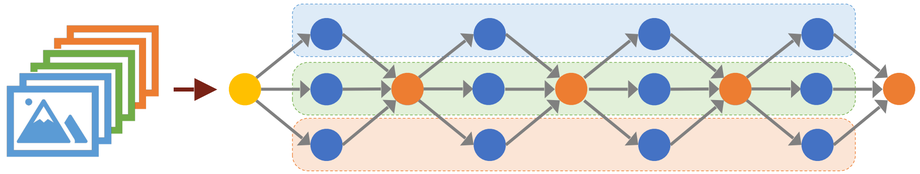
1. Data Parallelism:

- Split the data
- N copies of model
- high utilization, high memory cost, low communication
- Optimization: ZeRO 1/2/3, FSDP



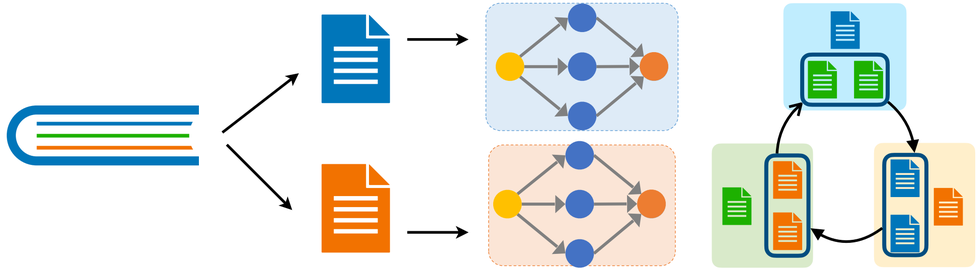
2. Pipeline Parallelism:

- Split the model by layer
- Single copy of model
- low utilization, low memory cost, medium communication



3. Tensor Parallelism:

- Split the model by tensor
- Single-copy of model
- high utilization, low memory cost, high communication



4. Sequence Parallelism:

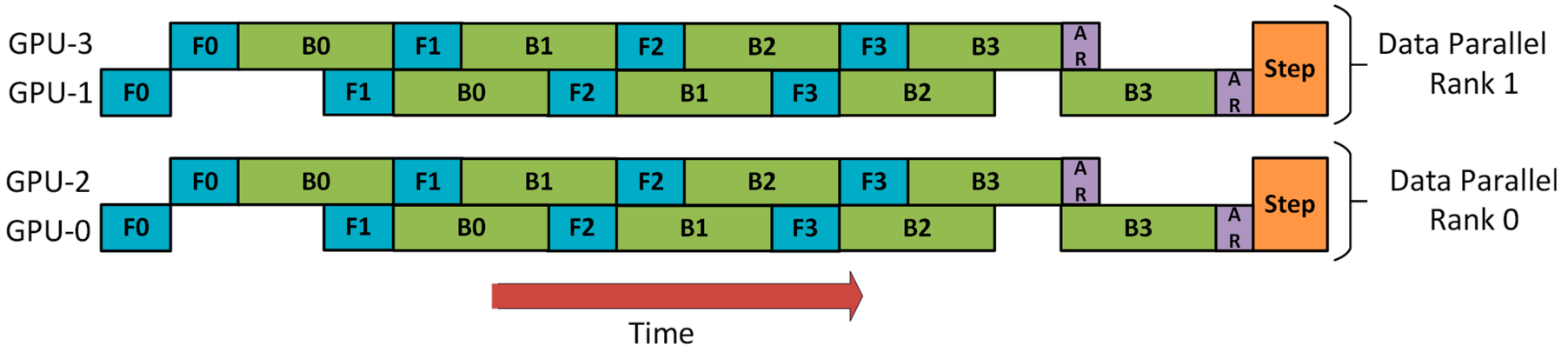
- Split the data by tokens
- Extra communication in Attention layers

- Infiniband:
  - EDR: 100Gb/s
  - HDR: 200Gb/s
  - NDR: 400Gb/s

Figures credit from CMU 15-849 [Jia 2022]

Hybrid Parallelism

2D Parallelism

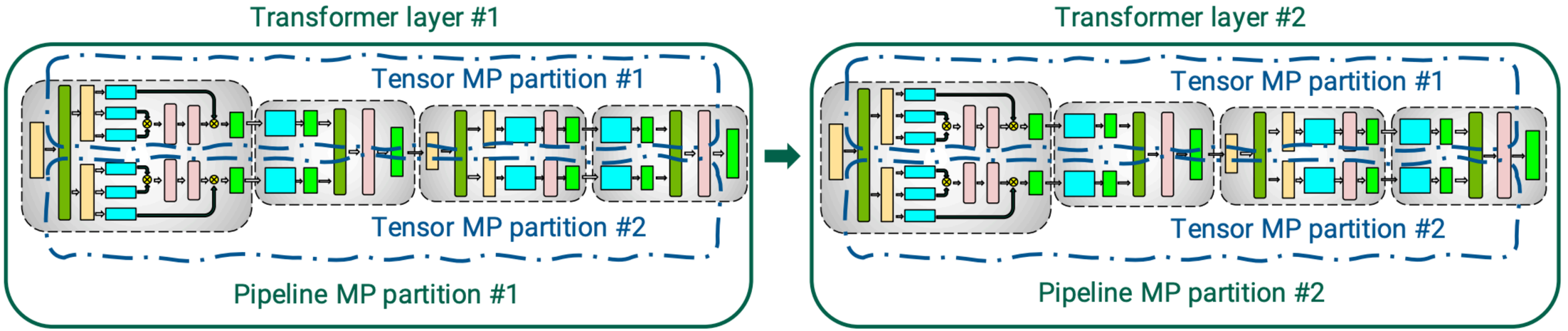


- Data Parallelism + Pipeline Parallelism
- Outer: [GPU1 & GPU3, GPU0 & GPU2] data parallelism.
- Inner:
  - [GPU1, GPU3] pipeline parallelism
  - [GPU0, GPU2] pipeline parallelism

https://www.deepspeed.ai/tutorials/pipeline/

Hybrid Parallelism

2D Parallelism



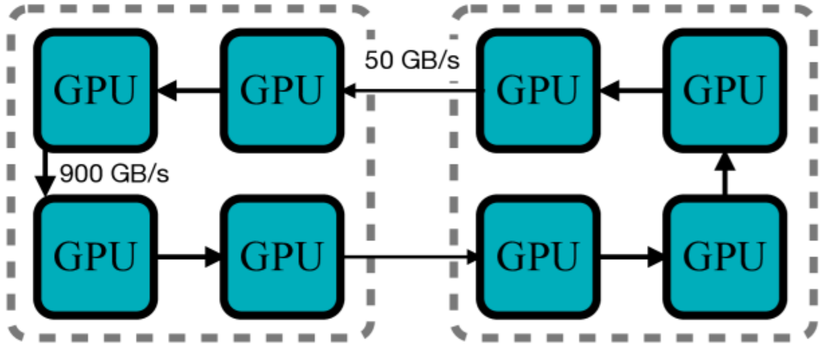
- Pipeline Parallelism + Tensor Parallelism
- Outer: Pipeline Parallelism
- Inner: Tensor Parallelism

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM

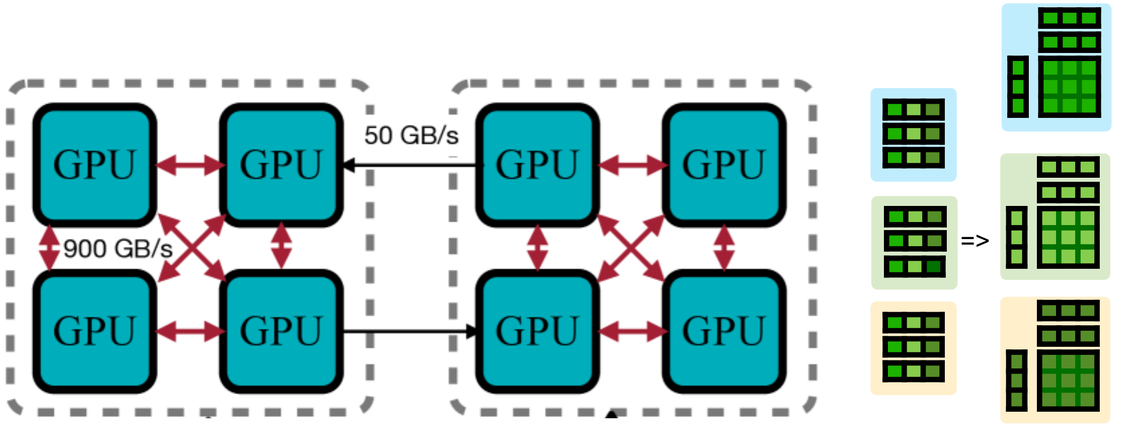
Hybrid Parallelism

2D Parallelism

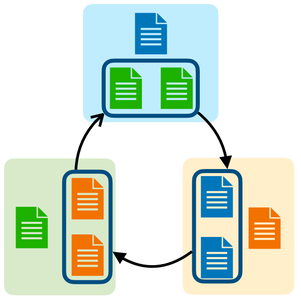
Under-utilization of intra-node bandwidth



1D Sequence Parallelism using Ring Attention



All-to-all re-partition within a node



Ring Attention between nodes

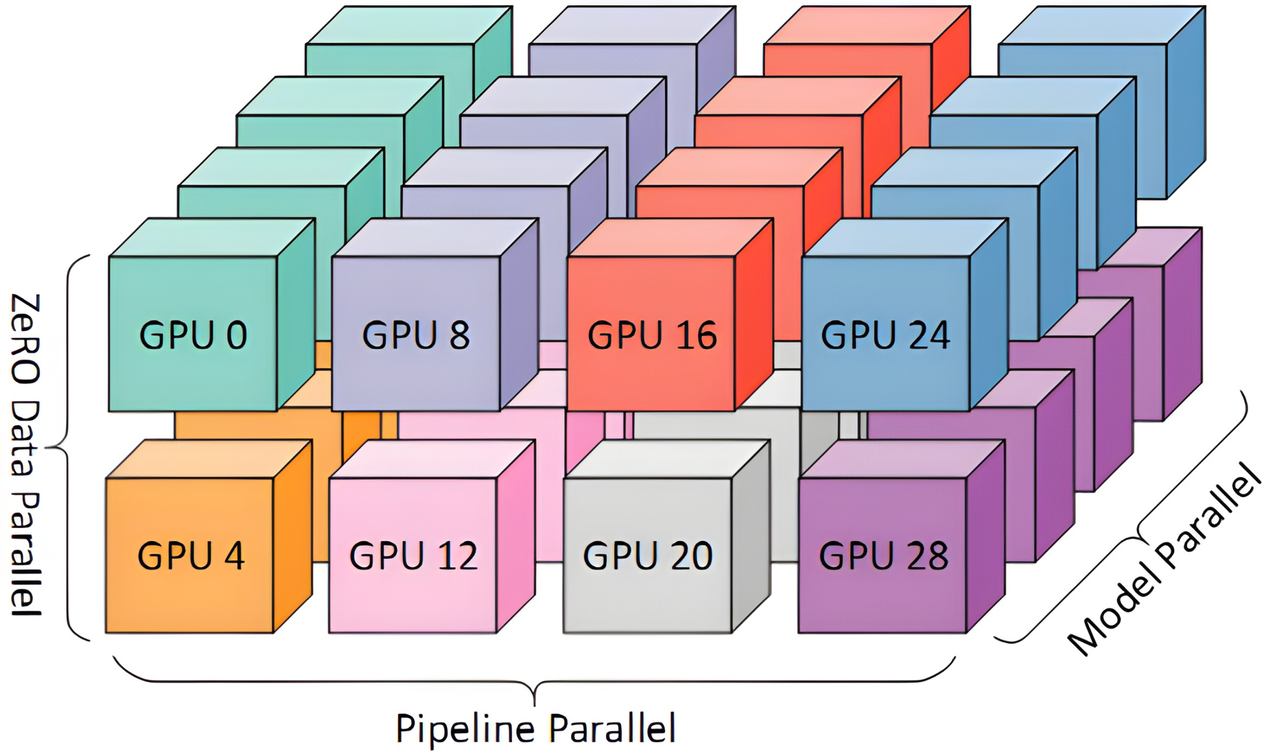
2D Attention in LongVILA

- 2D Sequence Parallelism
- Intra-node: all-to-all repartition
- Inter-node: ring attention

LongVILA: Scaling Long-Context Visual Language Models for Long Videos

Hybrid Parallelism

3D Parallelism



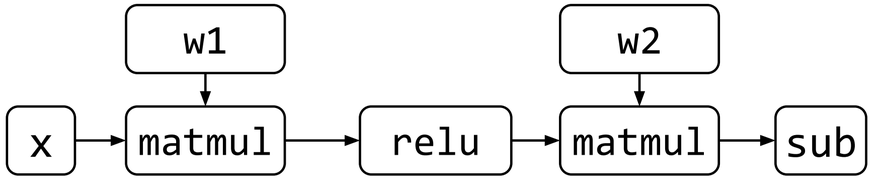
- Pipeline Parallelism + Tensor Parallelism + Data Parallel
- Same color denotes GPUs on the same server.

DeepSpeed: Extreme-scale model training for everyone

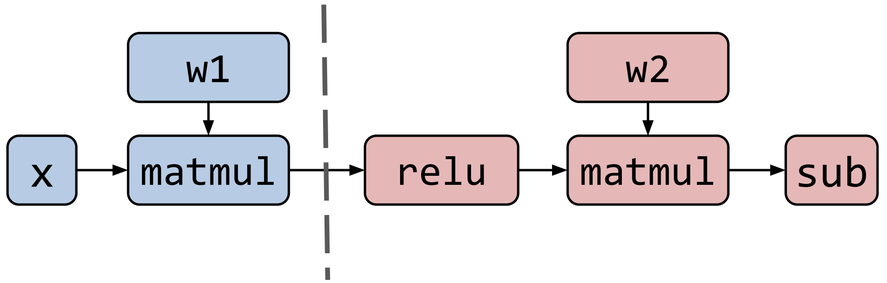
How to Auto Parallelize

Motivation

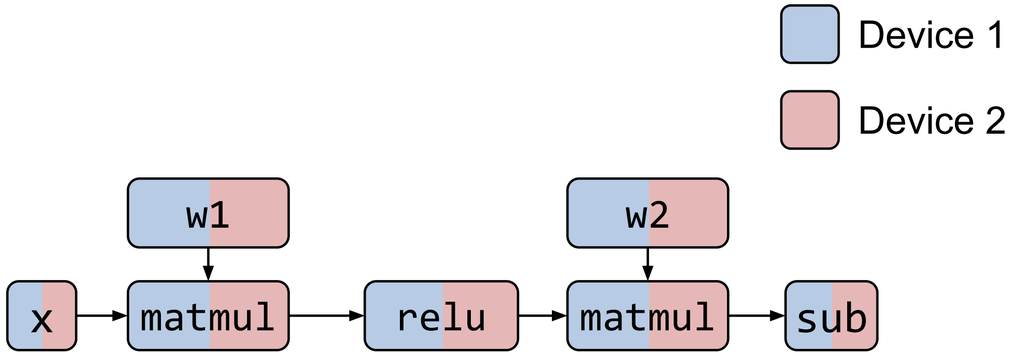
- When the model is too large to fit in GPU:
  - Split the model using pipeline parallelism
- When a layer is too large to fit in GPU:
  - Split the layer using tensor parallelism



(a) Computation Graph



(b) Pipeline Parallelism

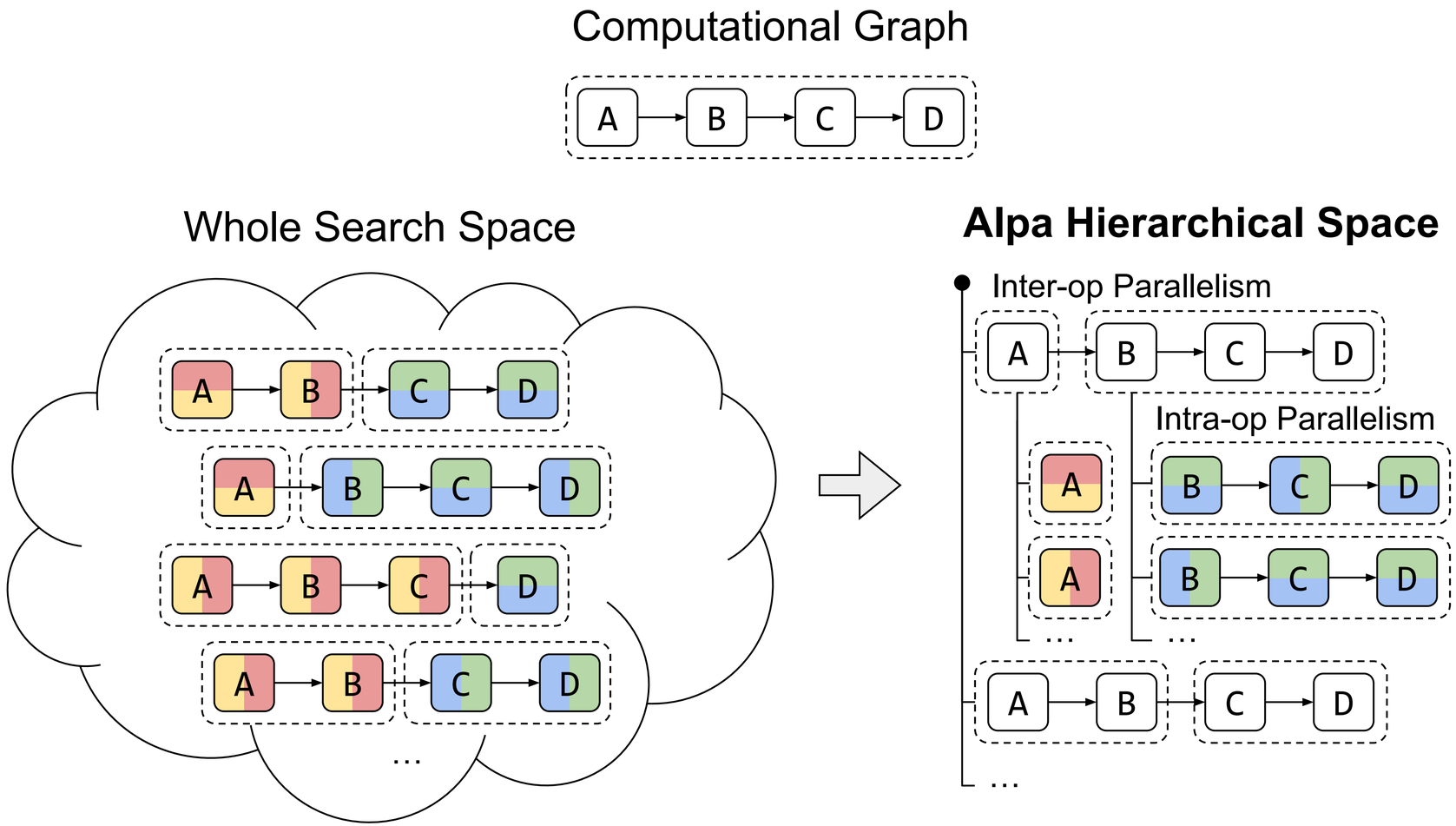


(c) Tensor Parallelism

But, how to find the best parallel strategies?

Alpa: A Unified Compiler for Distributed Training

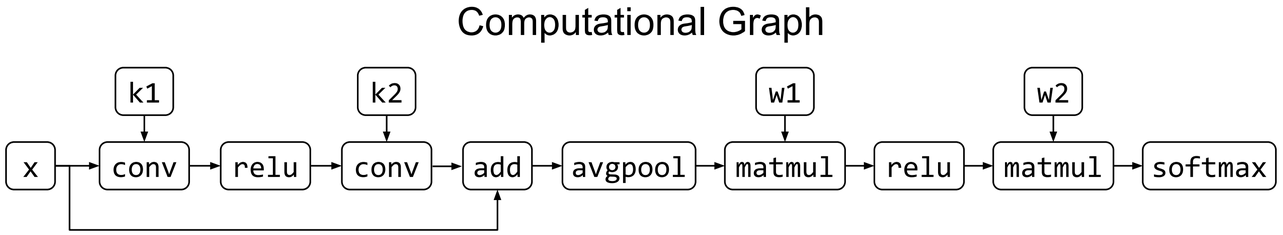
Define the Search Space for Parallel Strategies



Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning [Zheng et al. 2022]

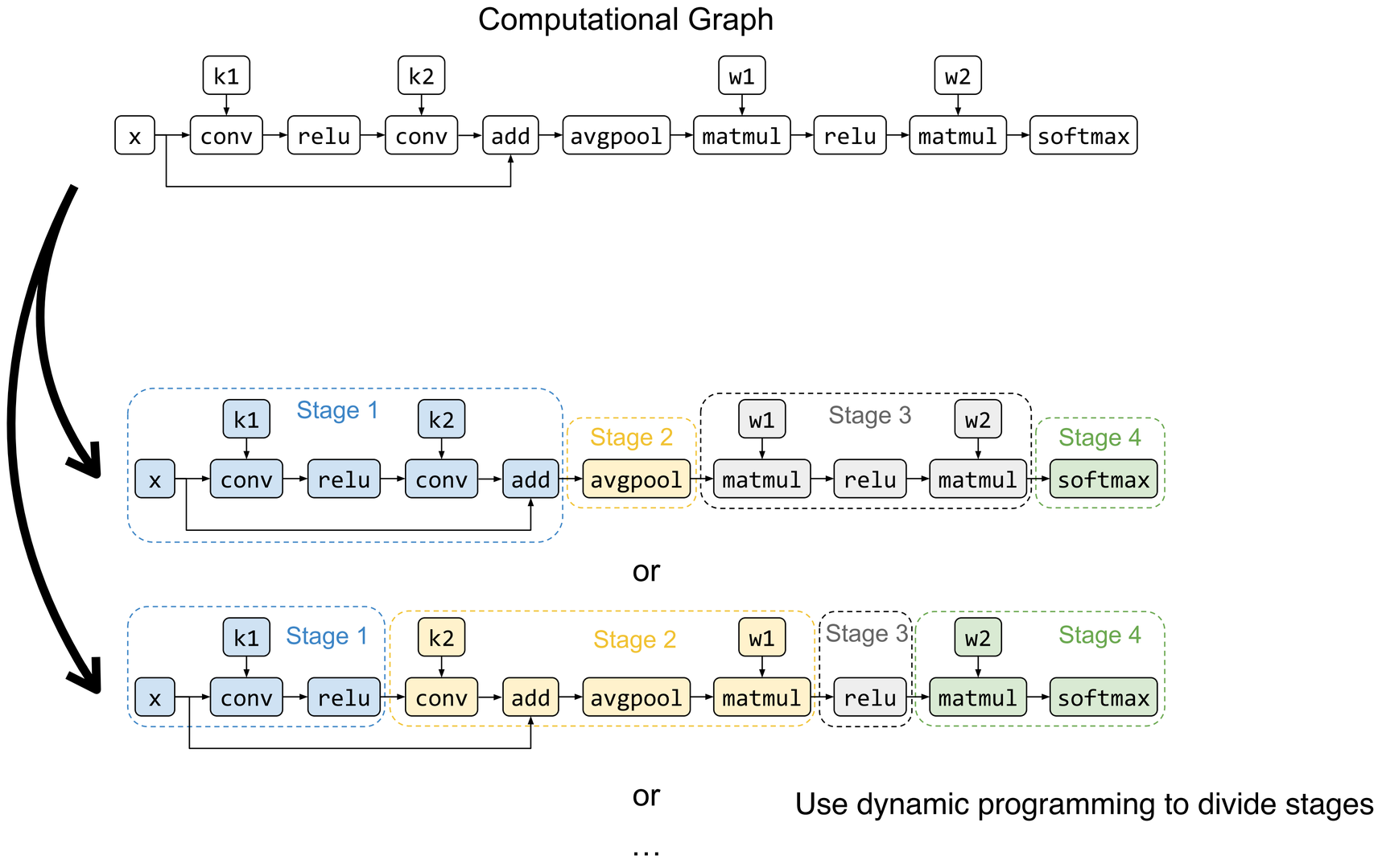
Alpa: A Unified Compiler for Distributed Training

Search for Inter-op Parallelism



Alpa: A Unified Compiler for Distributed Training

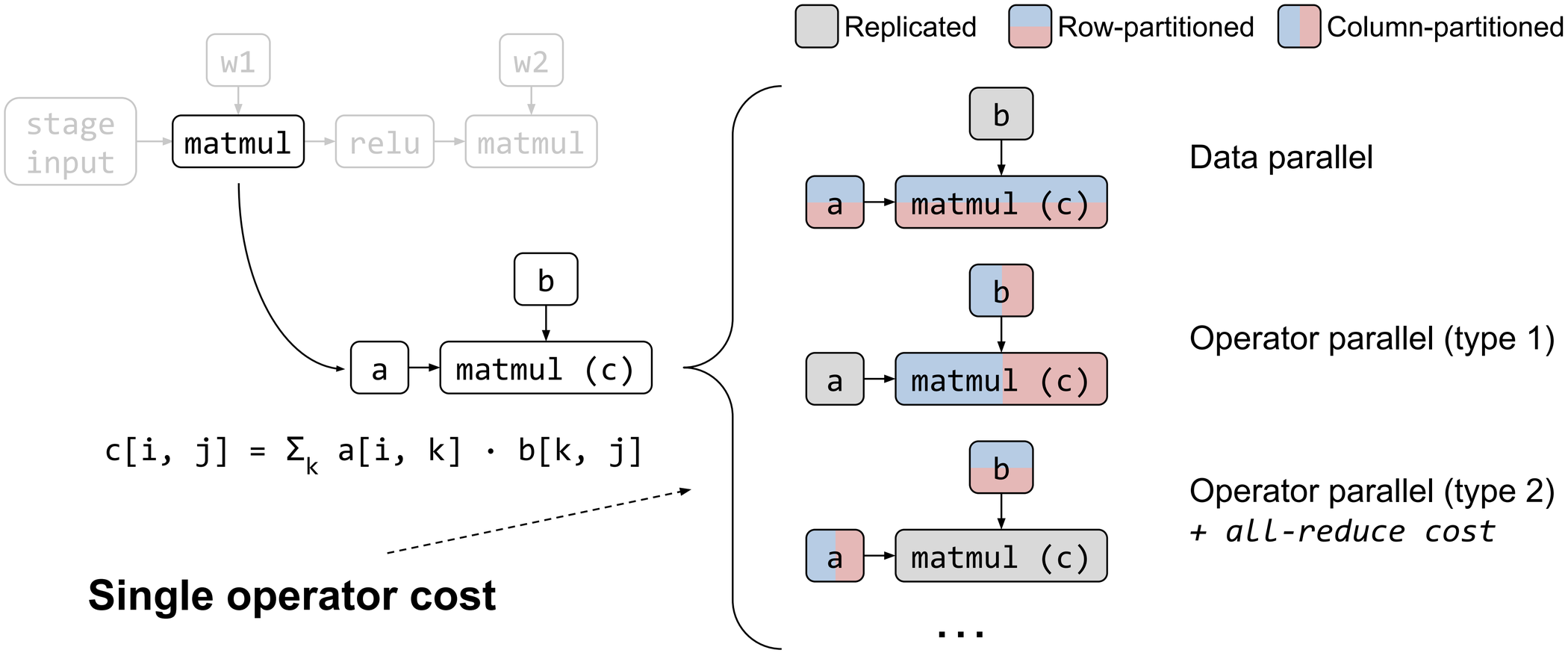
Search for Inter-op Parallelism



Use dynamic programming to divide stages

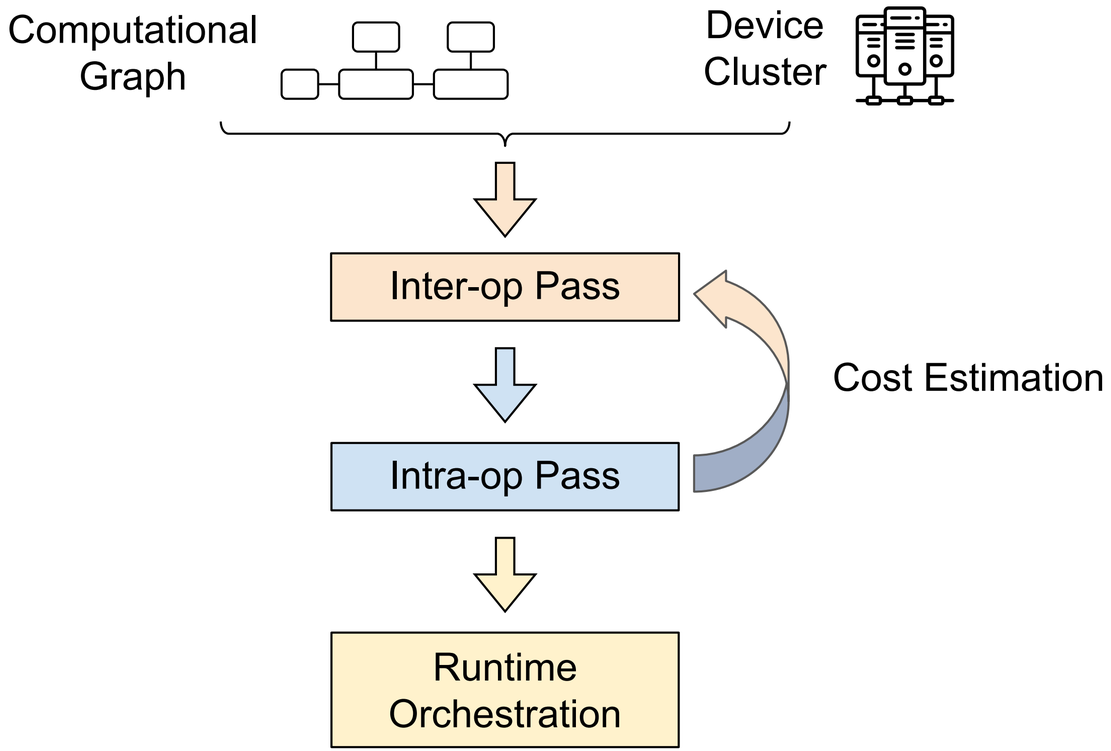
Alpa: A Unified Compiler for Distributed Training

Search for Intra-op Parallelism



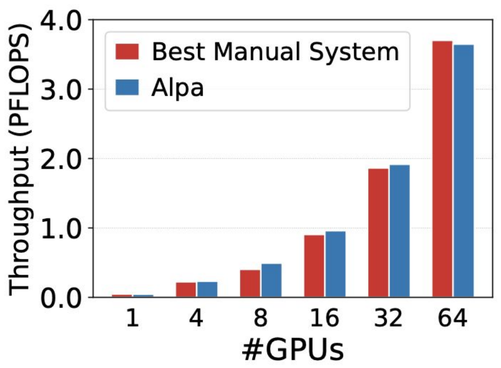
Cost = Compute Cost + Communication Cost + Resharding cost between ops
Use 0-1 linear programming to search parallel algorithm

Alpa: A Unified Compiler for Distributed Training



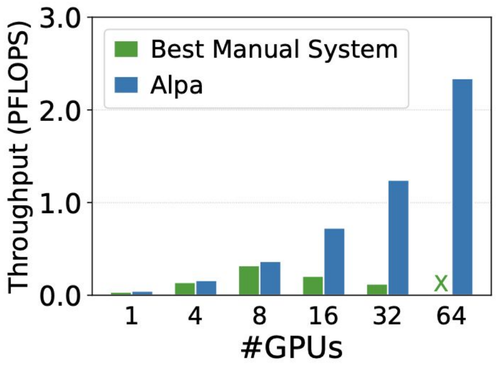
Optimize via **Automatic Search**, No need to manually config

GPT (up to 39B)



Match specialized manual systems.

GShard MoE (up to 70B)



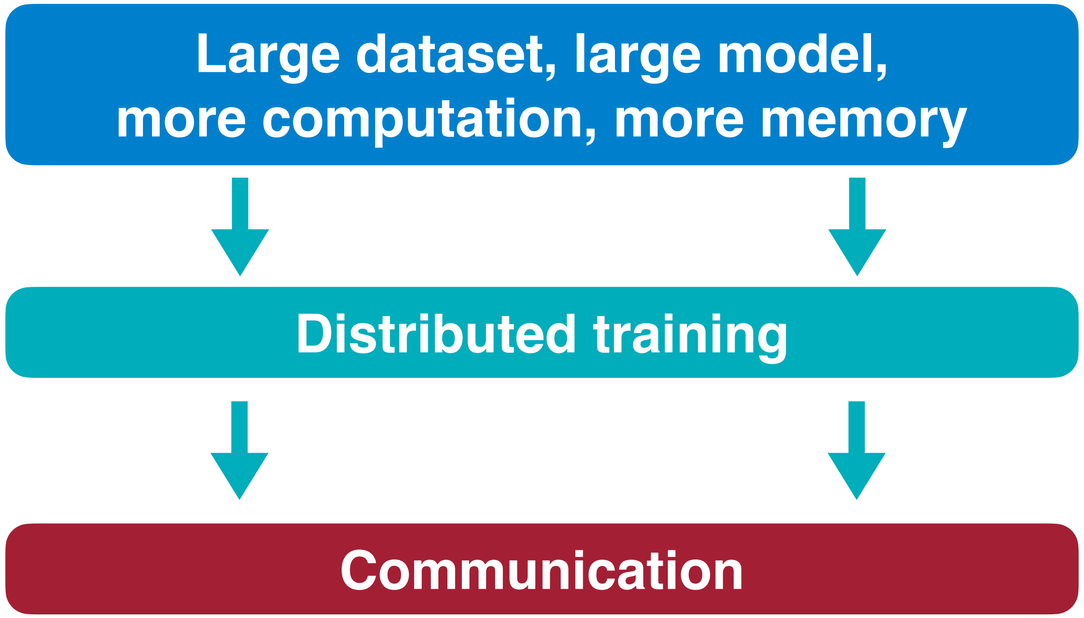
Outperform the manual baseline by up to 8x.

Lecture Plan

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. **Understand the bandwidth and latency bottleneck of distributed training**
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. Delayed gradient update: overcome the latency bottleneck

Bottlenecks of Distributed Training

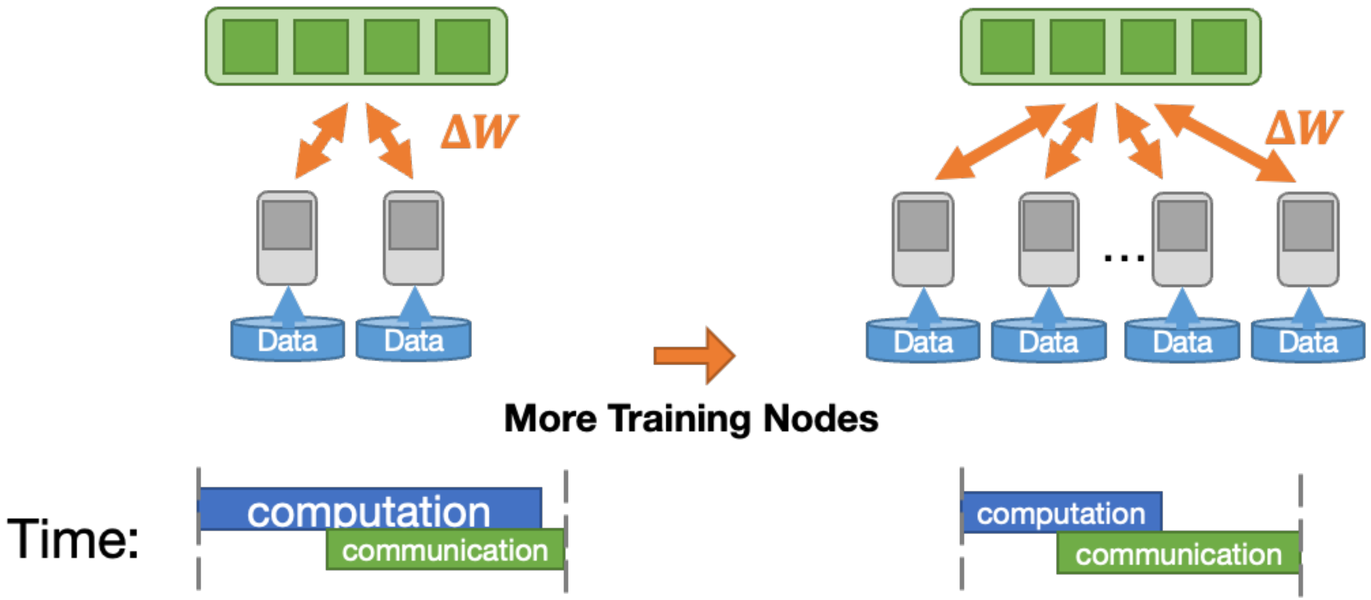
Communication is essential



Bottlenecks of Distributed Training

Communication

- Requires synchronization, high communication frequency
- Larger model, larger transfer size
- More training nodes, more communication (all-reduce), longer latency



Bandwidth-efficient deep learning [Han et al, 2018]

Distributed Training Scalability
under 20 Gbps Ethernet with GPUDirect™

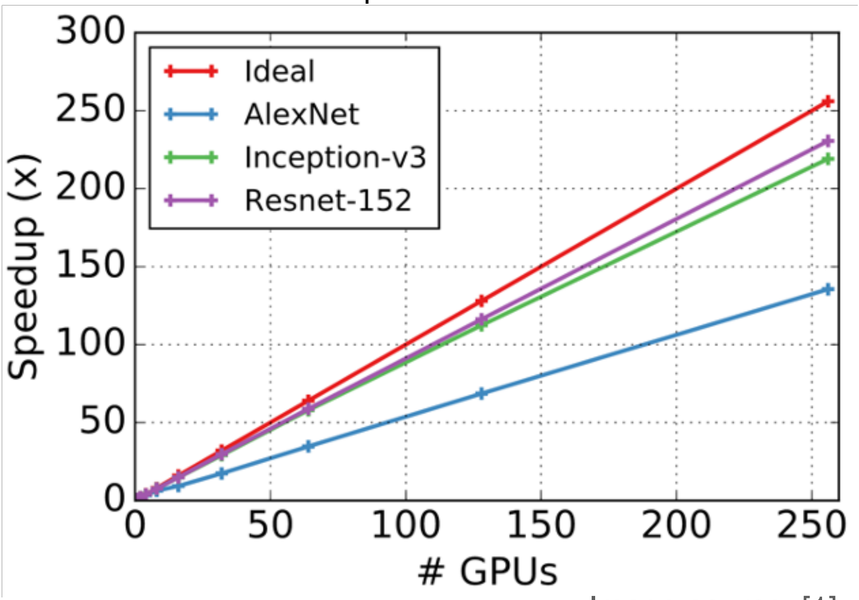
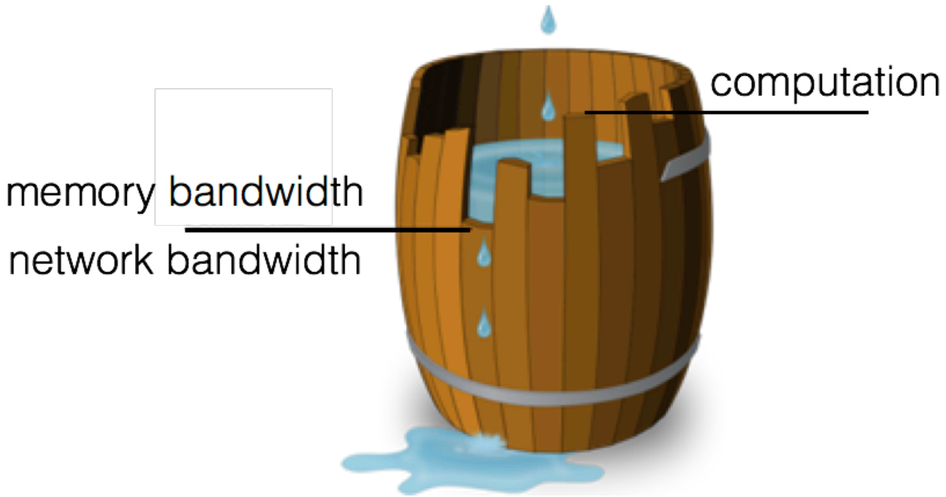


Image source: [1]

Bottlenecks of Distributed Training

Communication

- Requires synchronization, high communication frequency
- Larger model, larger transfer data size, longer transfer time
- More training nodes, more communication (all-reduce), longer latency
- **Networking bandwidth bottlenecks distributed training.**



Distributed Training Scalability
under 20 Gbps Ethernet with GPUDirect™

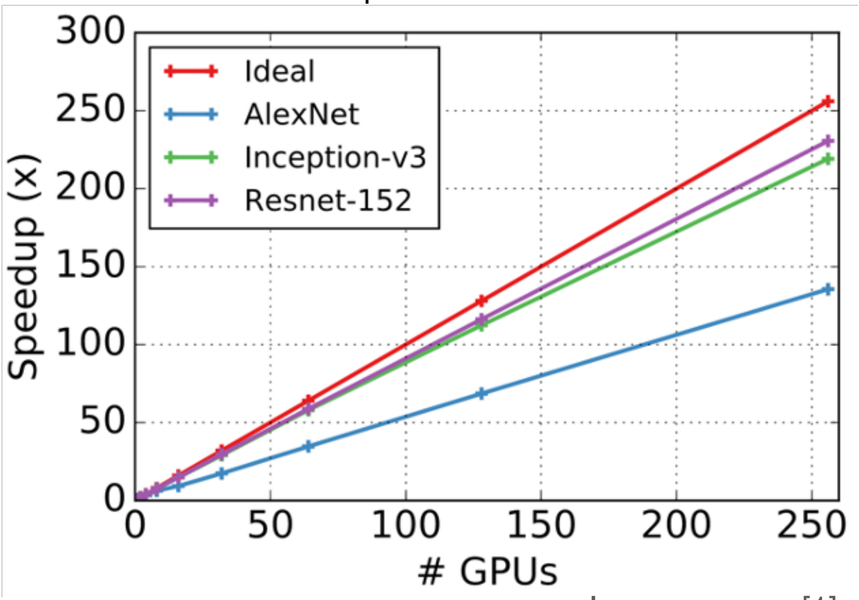
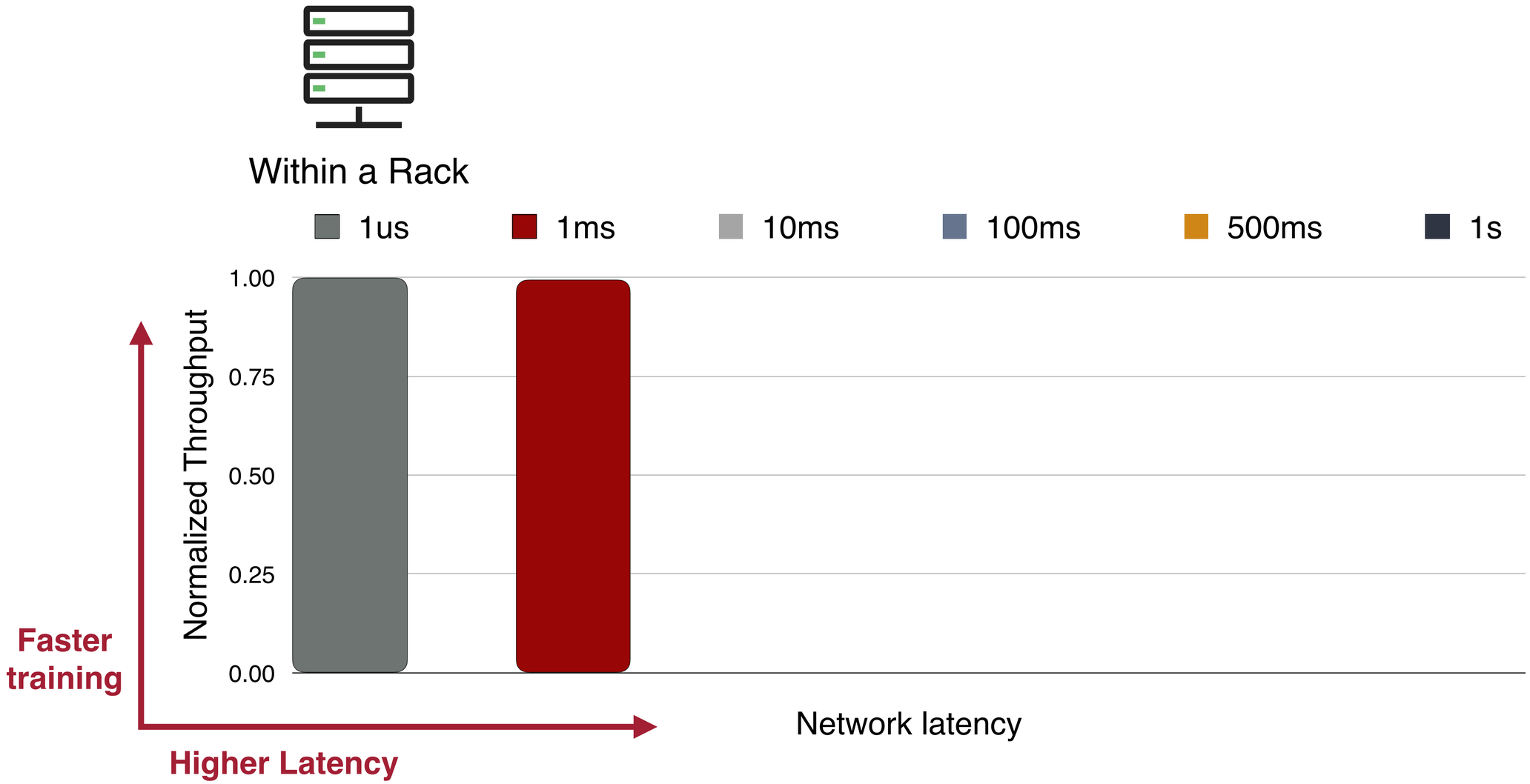


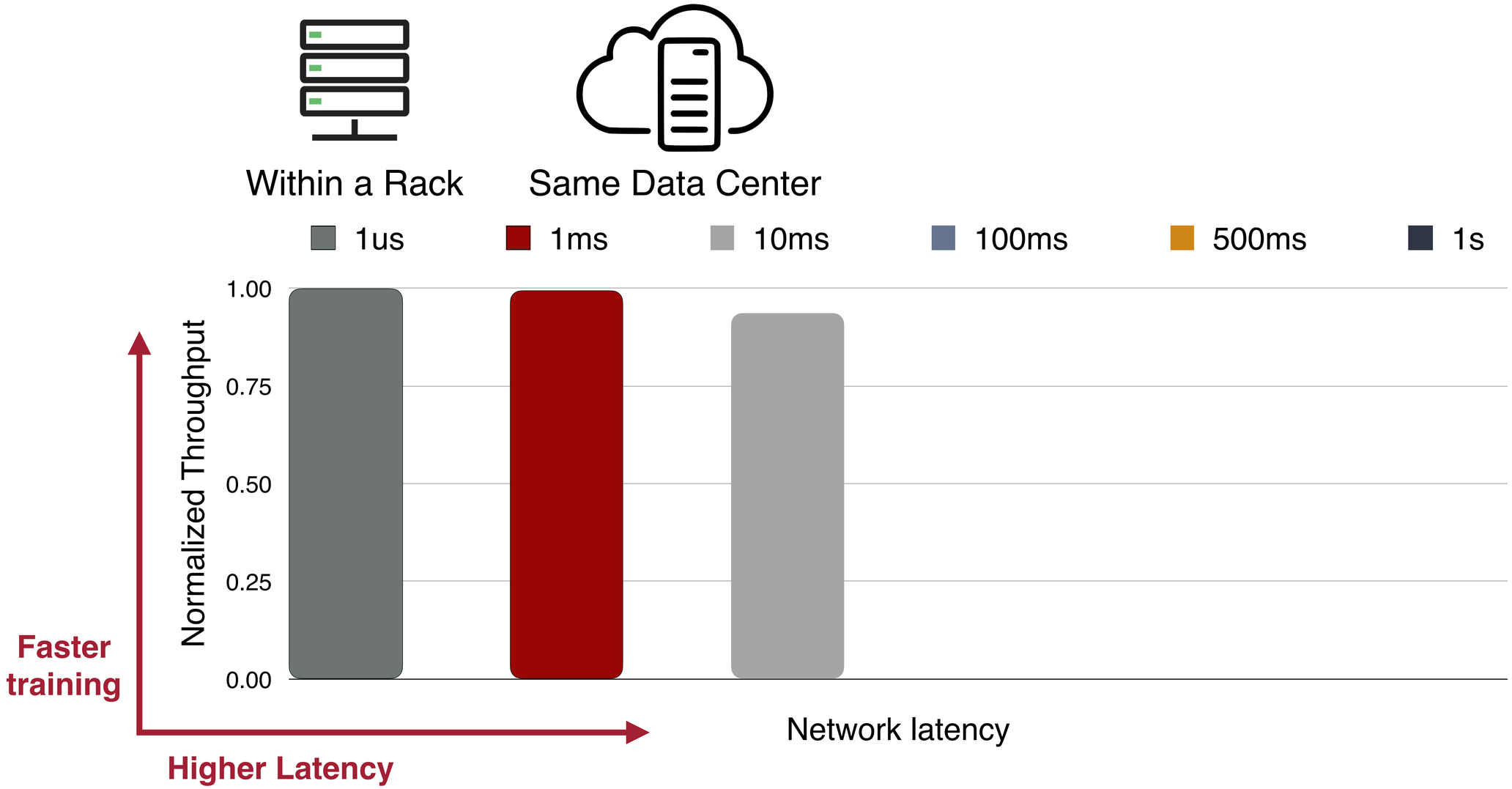
Image source: [1]

High Network Latency Slows Distributed Training



Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

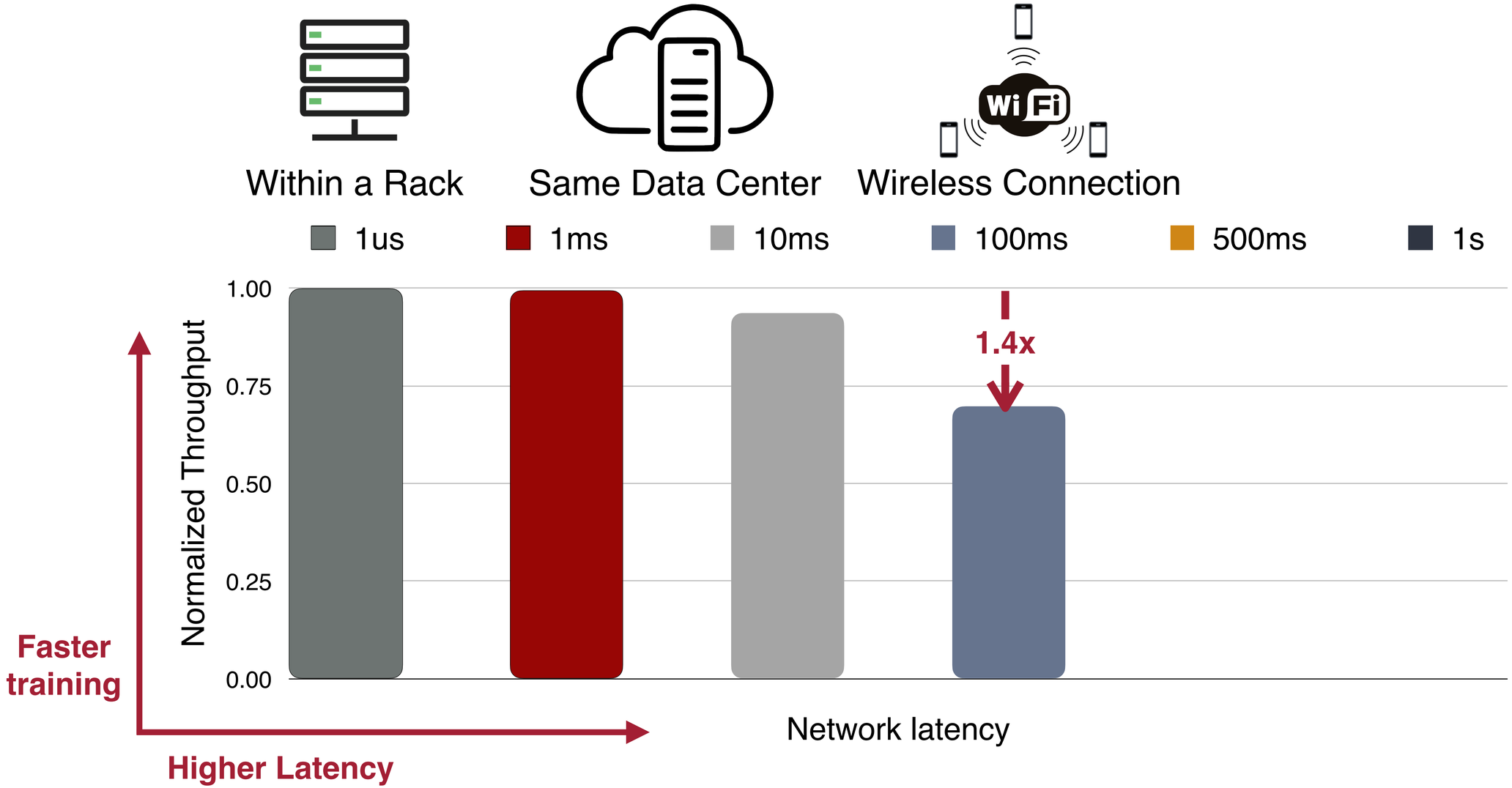
High Network Latency Slows Distributed Training



In cluster network latency does not affect training

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

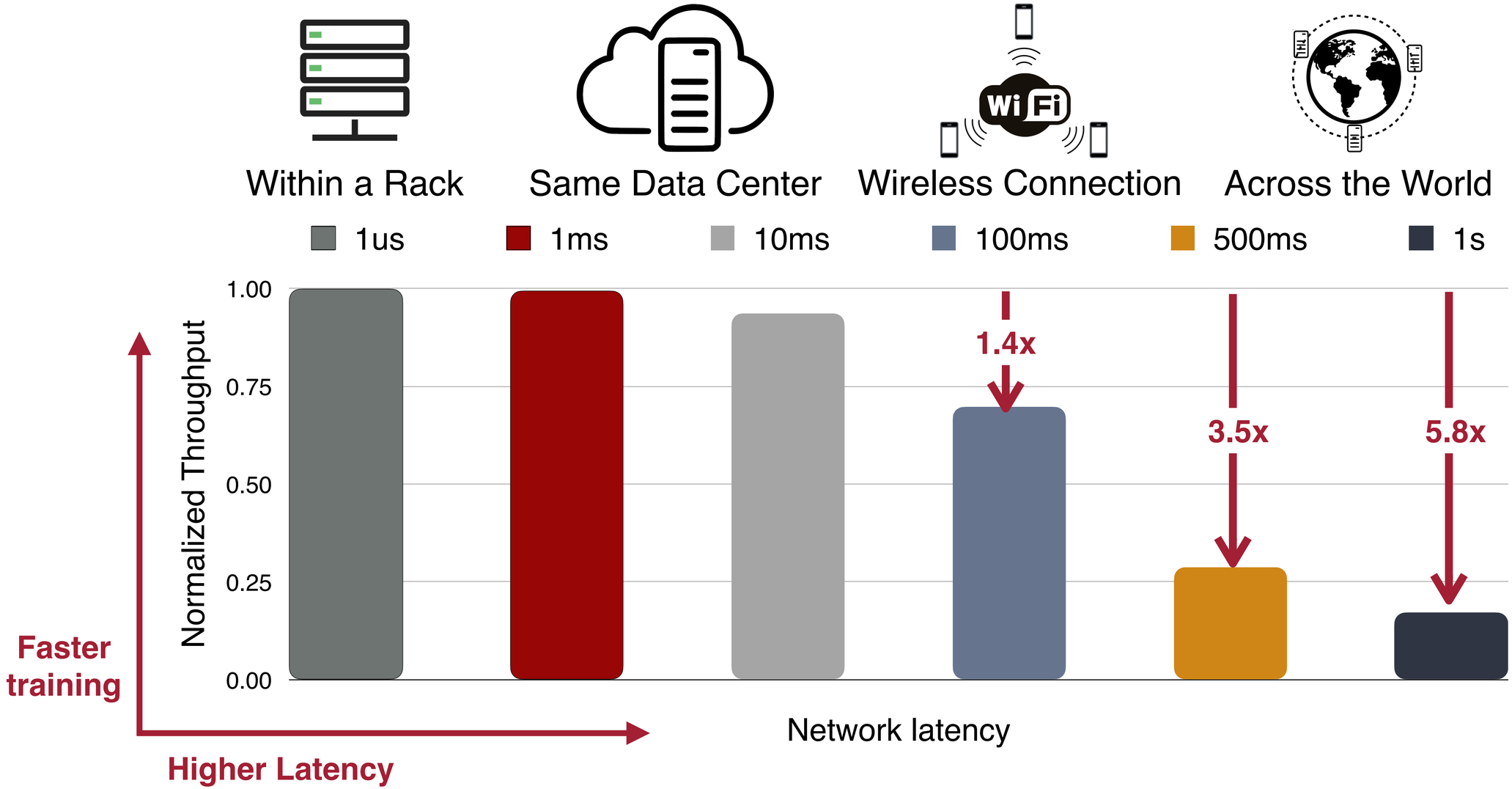
High Network Latency Slows Distributed Training



In home wireless connection slows the training by certain margin.

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

High Network Latency Slows Distributed Training



Long-distance connection slows the training by a large margin.

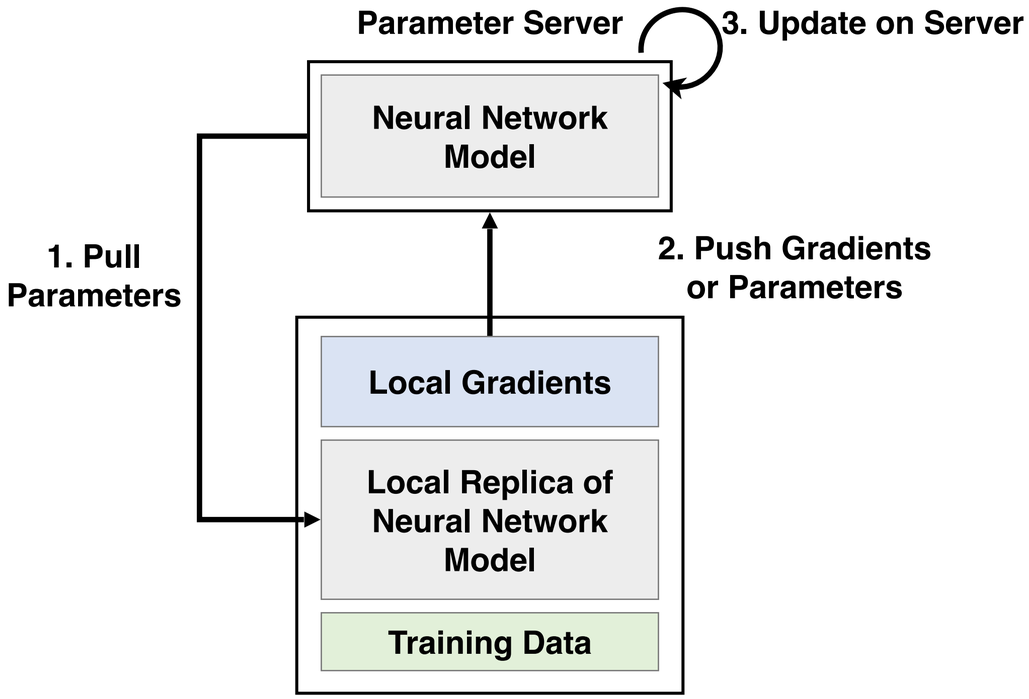
Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Lecture Plan

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. Understand the bandwidth and latency bottleneck of distributed training
3. **Gradient compression: overcome the bandwidth bottleneck**
    1. **Gradient Pruning: Sparse Communication, Deep Gradient Compression**
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. Delayed gradient update: overcome the latency bottleneck

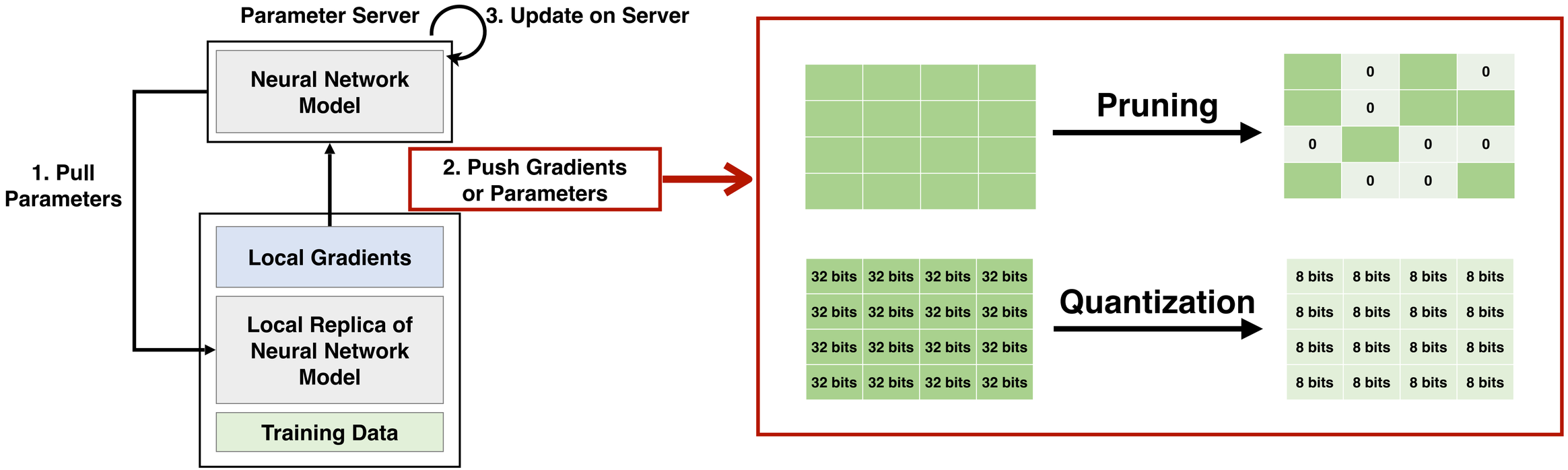
Reduce Transfer Data Size

Recall the workflow of Parameter-Server Based Distributed Training



Reduce Transfer Data Size

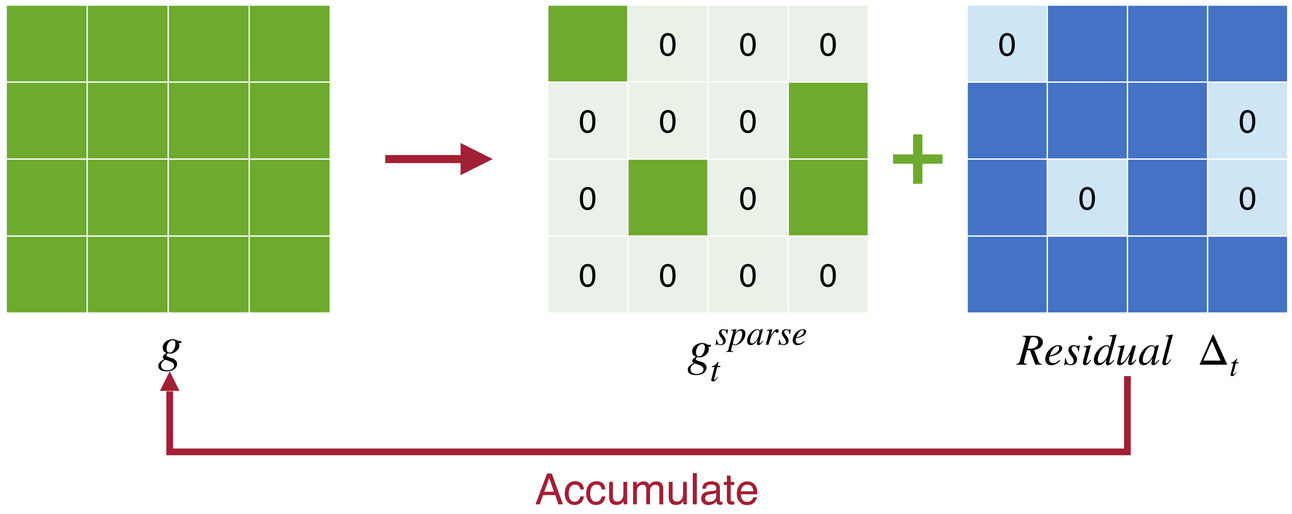
Recall the workflow of Parameter-Server Based Distributed Training



Sparse Communication

Gradient pruning with local gradient accumulation

- Only send **top-k** gradients (by magnitude)
- Keep the un-pruned part as **error feedback (residual)**



Sparse communication for distributed gradient descent [Alham Fikri et al 2017]
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Sparse Communication

Gradient pruning with local gradient accumulation

- Only send **top-k** gradients (by magnitude)
- Keep the un-pruned part as **error feedback (residual)**
- Improve training speed

Drop Ratio	words/sec (NMT)	images/sec (MNIST)
0%	13100	2489
90%	14443	3174
99%	14740	3726
99.9%	14786	3921

- Experiment on 4 Nvidia Titans

Sparse communication for distributed gradient descent [Alham Fikri et al 2017]

Sparse Communication

Gradient pruning with local gradient accumulation

- Only send **top-k** gradients (by magnitude)
- Keep the un-pruned part as **error feedback (residual)**
- Improve training speed
- Work for simple neural networks, but fail on modern models like ResNet

Setting	Top 1 Accuracy of ResNet-110 on Cifar10		
GPUs=8 bs=128	Baseline	93.75%	0
	Sparse Communication	**92.75%**	**-1.00%**

Sparse communication for distributed gradient descent [Alham Fikri et al 2017]
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Limitations of Sparse Communication

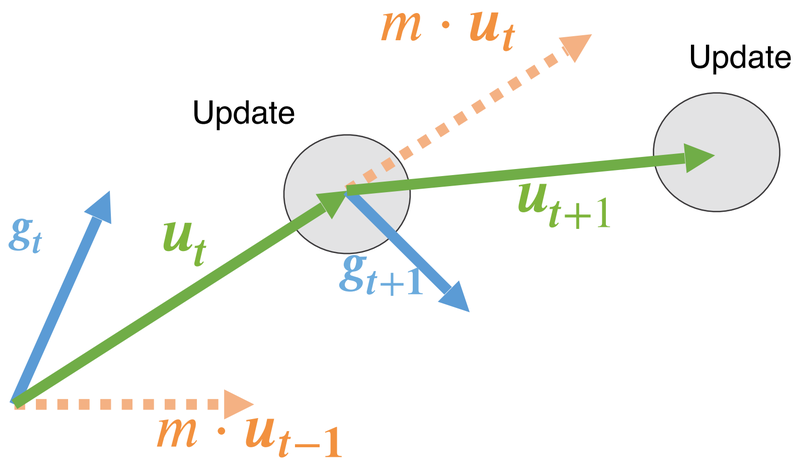
Why performance degradation in gradient dropping?

Momentum!

Optimizers with Momentum

- Momentum is an extension to the gradient descent, useful to overcome local minima and oscillation of noisy gradients.

- Momentum is widely used to train modern deep neural networks (used in AlexNet, VGG, ResNet, Transformer, BERT …)



velocity: u_t

momentum m

Vanilla Momentum SGD

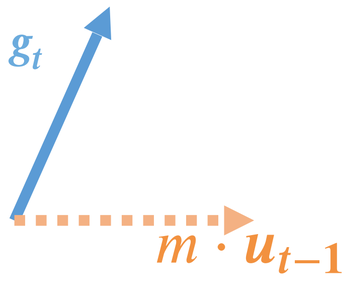
$$u_t = m \cdot u_{t-1} + g_t$$

$$w_t = w_{t-1} - \eta \cdot u_t$$

On the momentum term in gradient descent learning algorithms. [Qian 1999]
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Optimizers with Momentum

Step 1 - Obtain the gradients g_t



Vanilla Momentum SGD

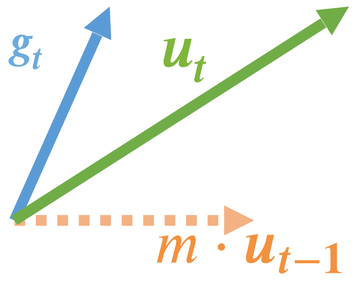
$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Calculate g_t same as the vanilla SGD.

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Optimizers with Momentum

Step 2 - Calculate the velocity u_t using momentum m



Vanilla Momentum SGD

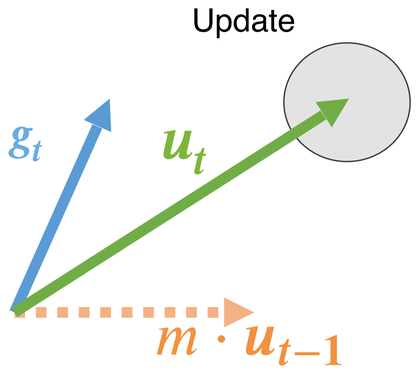
$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Momentum m is usually a value between 0 and 1.

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Optimizers with Momentum

Step 3 : Update model weights using velocity u_t



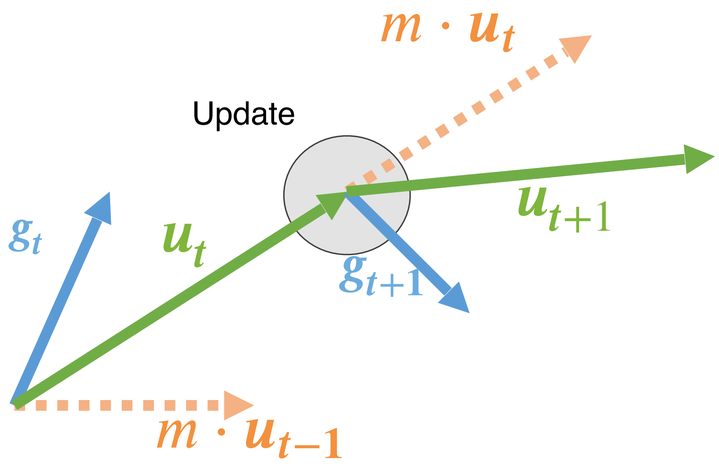
Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Optimizers with Momentum

Repeat, calculate the velocity for the next iteration



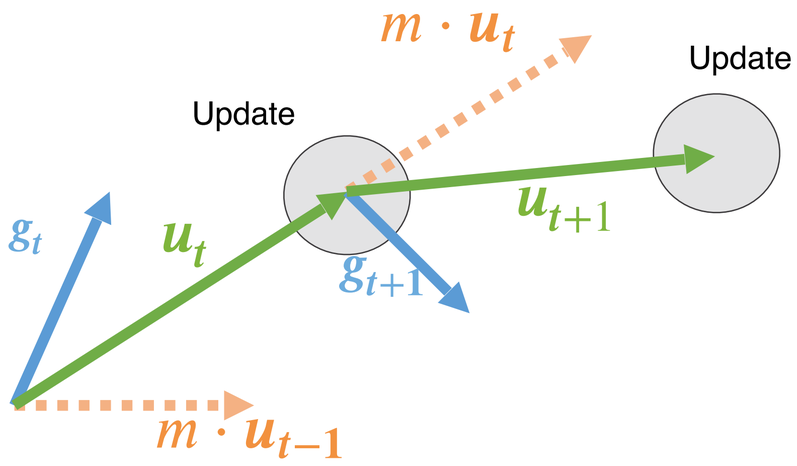
Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Optimizers with Momentum

Repeat, update weights



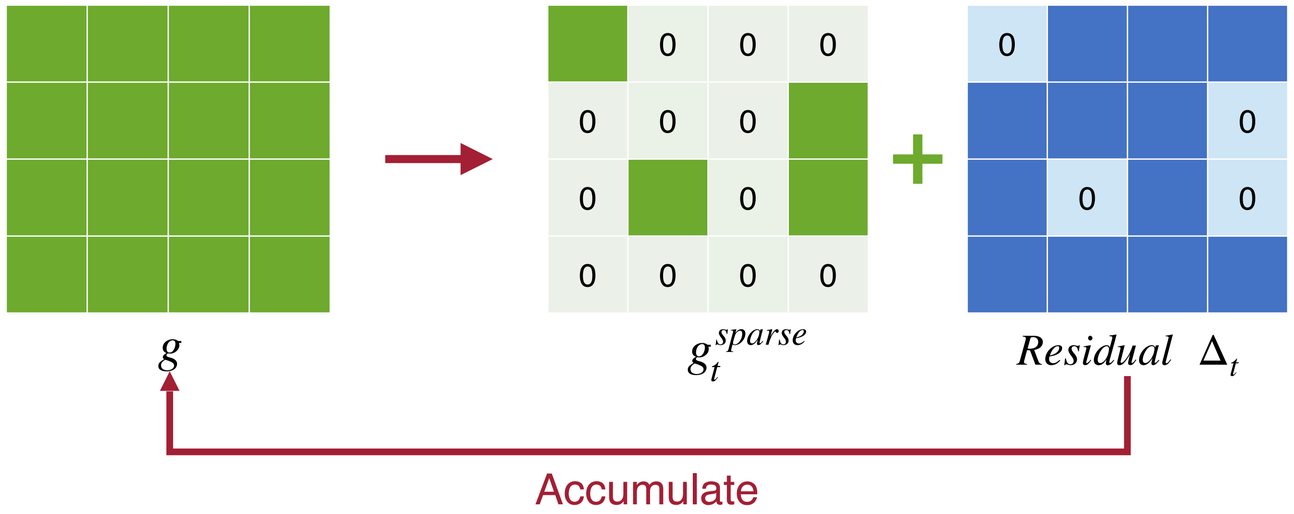
Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression w/o Momentum Correction

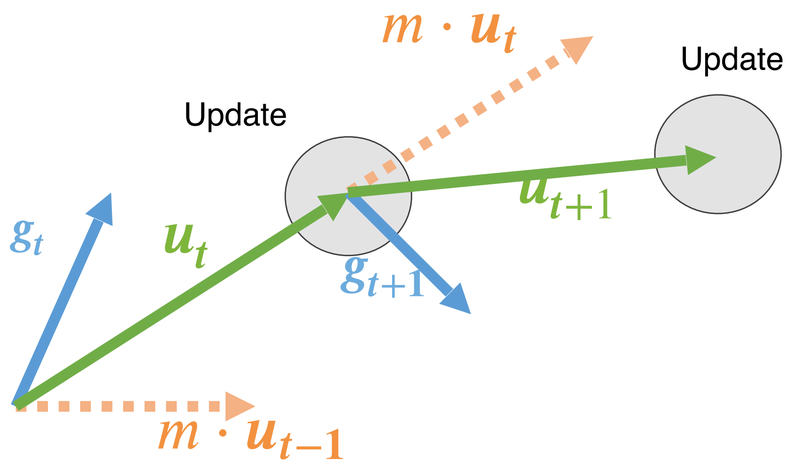
- In sparse communication, the pruned gradients are accumulated in local buffers.
- What will happen if we directly use the accumulated gradients for momentum calculation?



Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

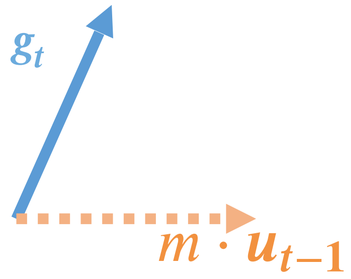
Deep Gradient Compression w/o Momentum Correction

Gradient g_t, velocity u_t, momentum m



Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

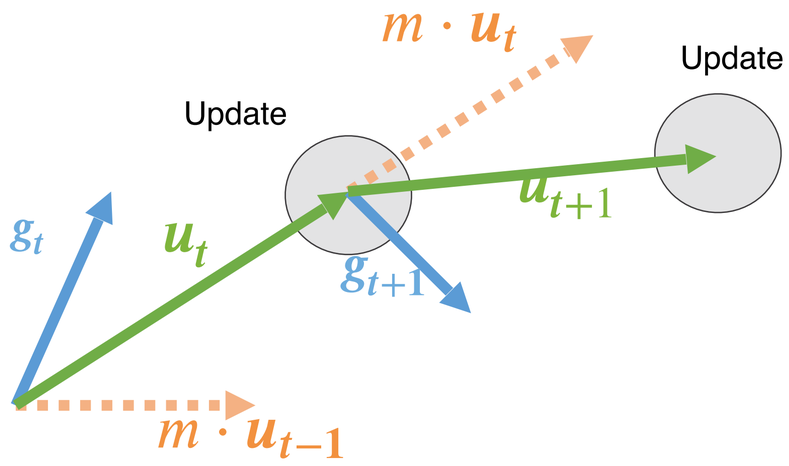


Momentum SGD with Local Accumulation

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

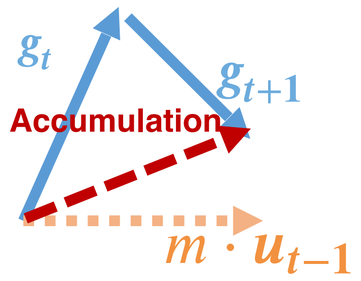
Deep Gradient Compression w/o Momentum Correction

Accumulate the gradients in the local buffer



Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

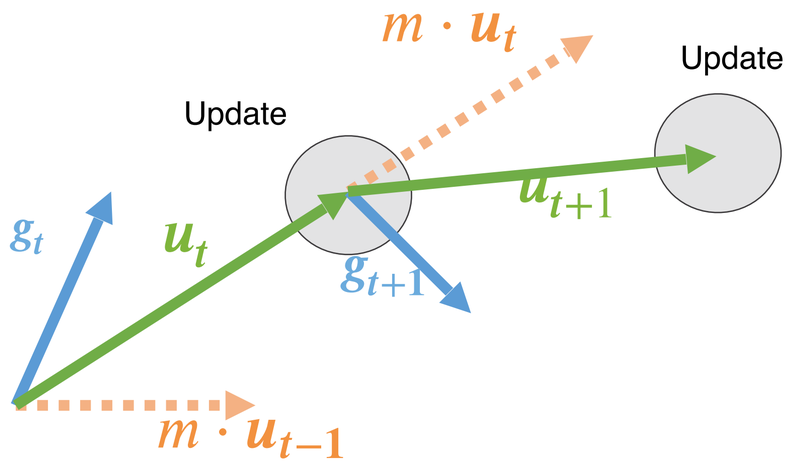


Momentum SGD with Local Accumulation

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

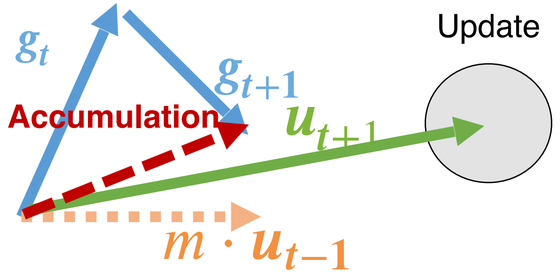
Deep Gradient Compression w/o Momentum Correction

Compute the velocity by adding accumulated gradient and momentum



Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

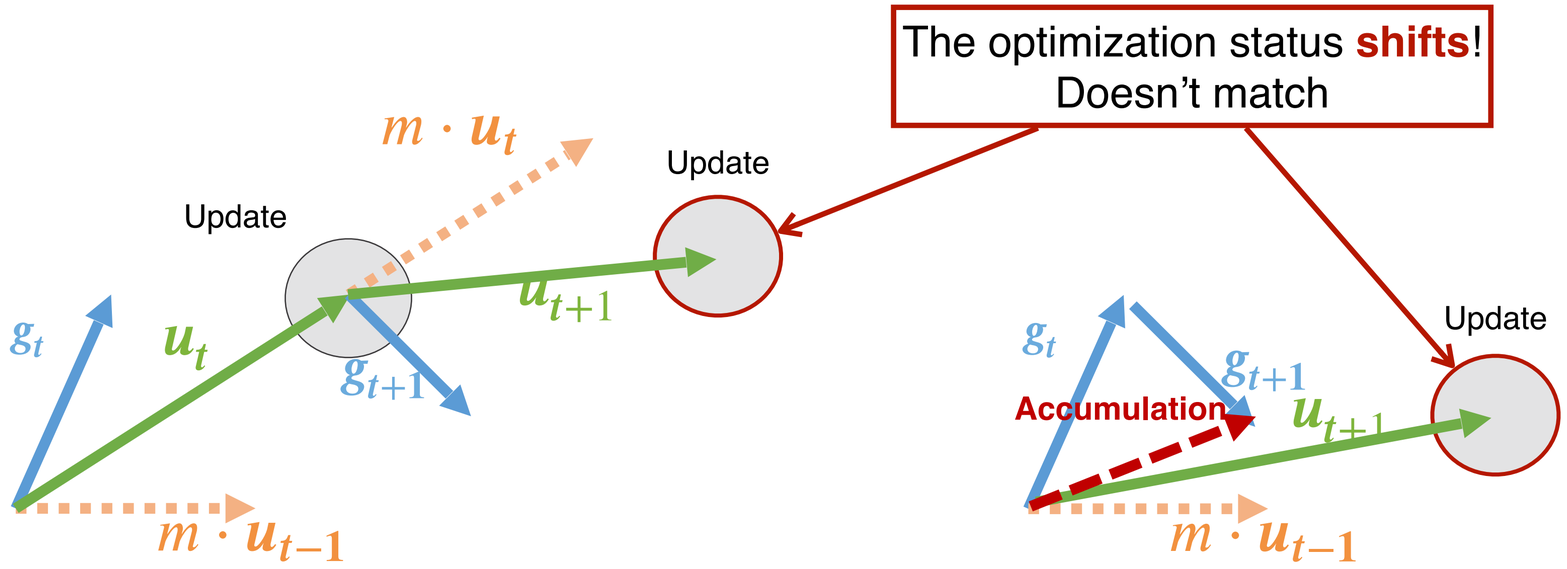


Momentum SGD with Local Accumulation

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression w/o Momentum Correction

the optimization trajectories doesn't match!



Vanilla Momentum SGD

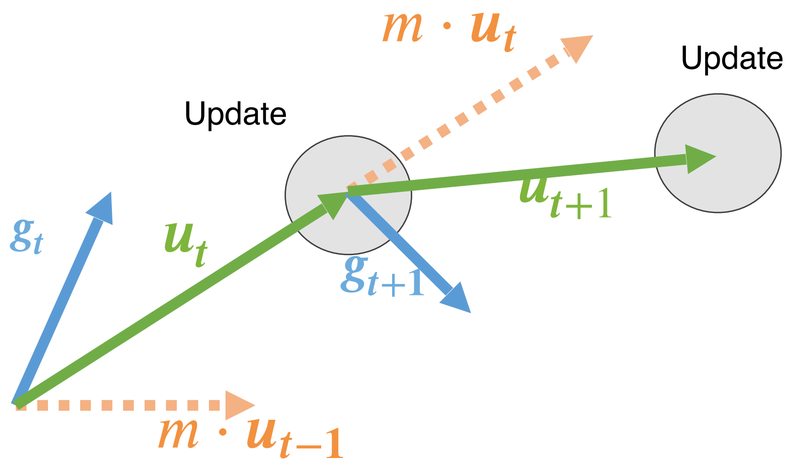
$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$

Momentum SGD with Local Accumulation

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

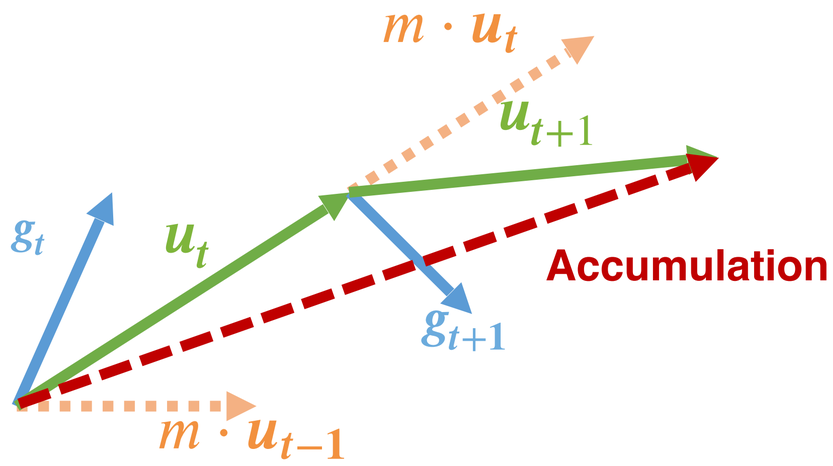
Deep Gradient Compression w/ Momentum Correction

Momentum Correction



Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$



Momentum Correction

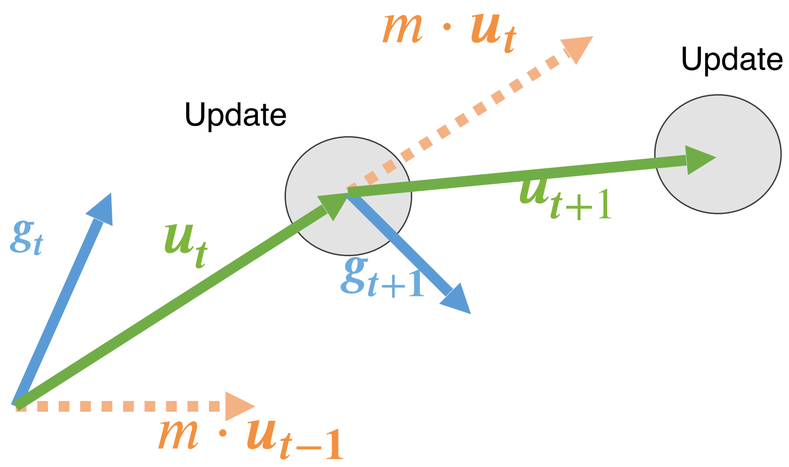
Accumulate the velocity, not gradients

accumulate the velocity through iterations

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

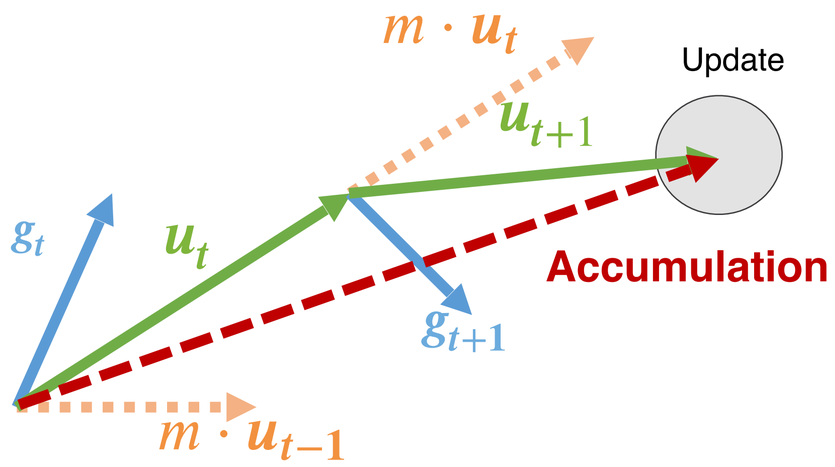
Deep Gradient Compression w/ Momentum Correction

We should accumulate the velocity, not the gradient



Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$



Momentum Correction

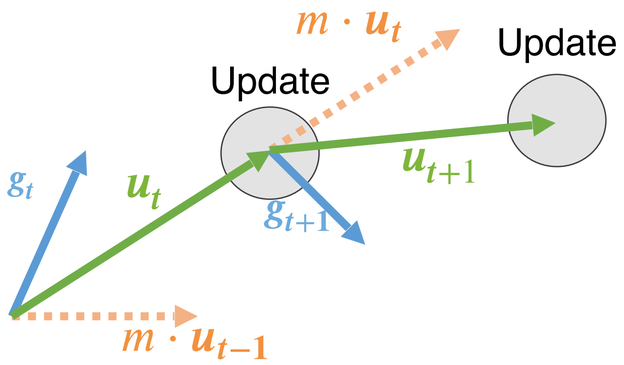
Accumulate the velocity, not gradients

the optimization status now matches vanilla momentum SGD.

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

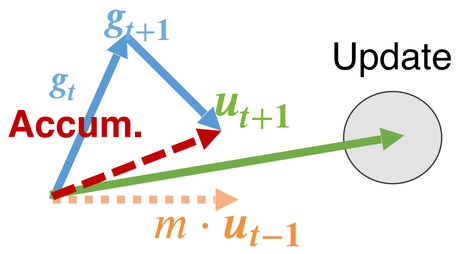
Deep Gradient Compression w/ Momentum Correction

Summary: momentum correction

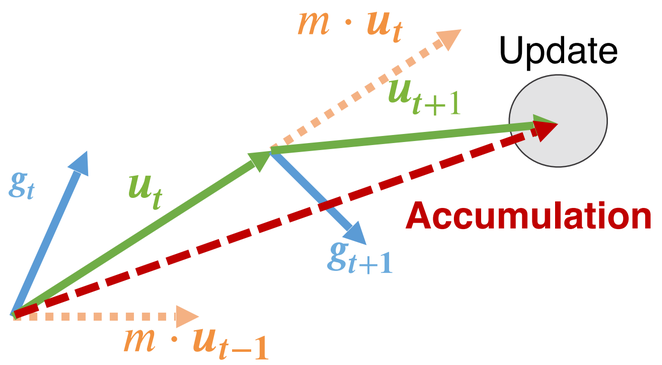


Vanilla Momentum SGD

$$u_t = m \cdot u_{t-1} + g_t$$
$$w_t = w_{t-1} - \eta \cdot u_t$$



Without Momentum Correction
(Inaccurate Direction)



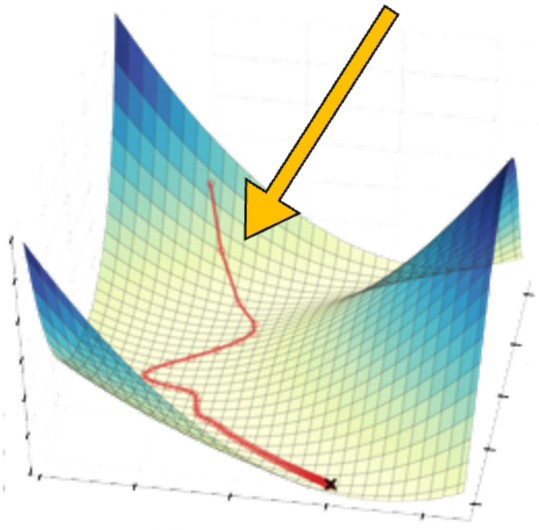
With Momentum Correction
(Correct Direction)

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Warm Up Training

- In the **early stages** of training, the network is **changing rapidly**
- Local gradient accumulation and stale gradient will **aggravate** the problem

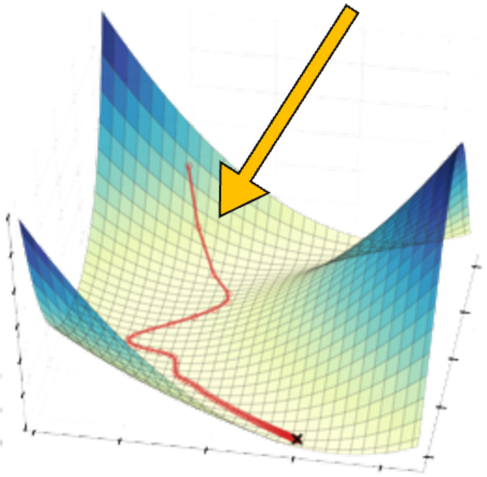


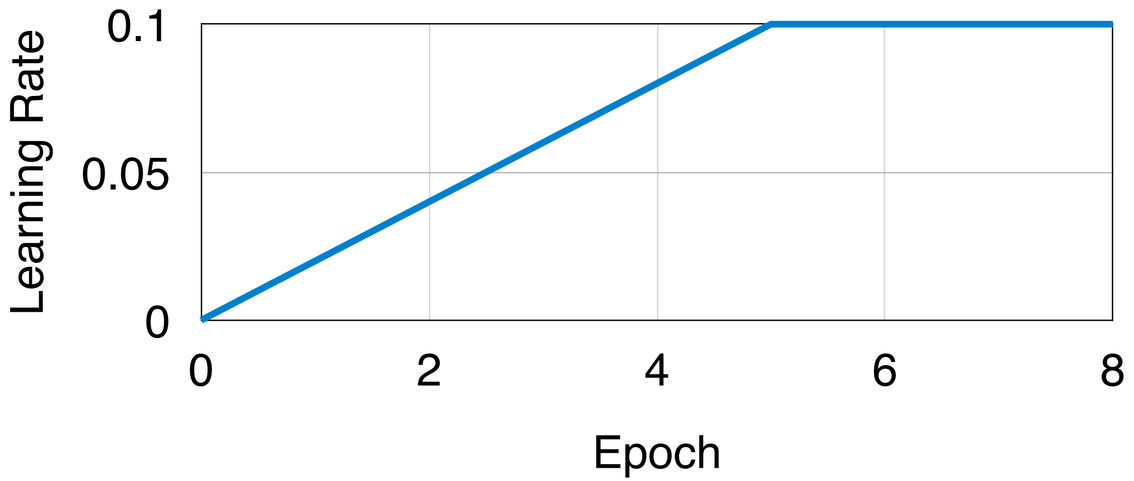
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Warm Up Tricks

- **Warm up the learning rate**
- **Warm up sparsity**
  - **avoid** a sudden change in sparsity
  - **exponentially** increasing sparsity in first several epochs → help optimizer adapt to larger sparsity



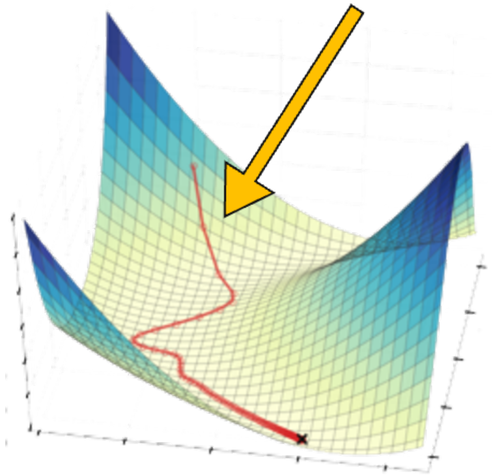


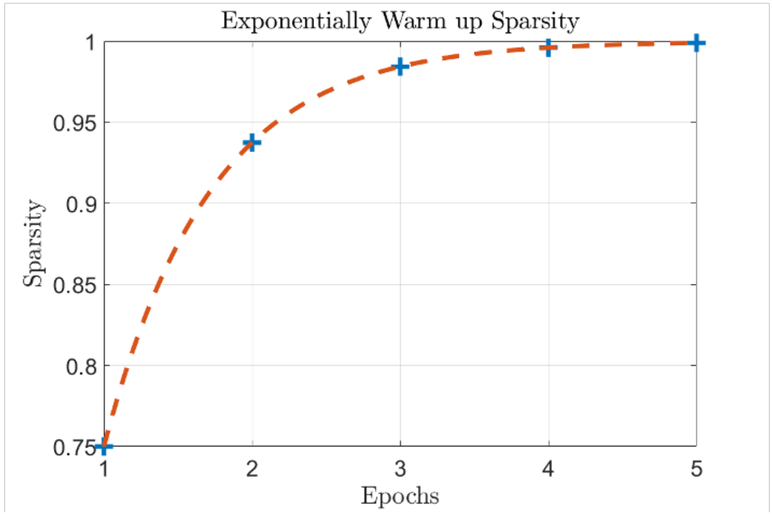
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Warm Up Tricks

- Warm up learning rate
- **Warm up sparsity**
  - **avoid** a sudden change in sparsity
  - **exponentially** increasing sparsity in first several epochs → help the optimizer adapt to larger sparsity





Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

DGC with 99.9% sparsity still converges

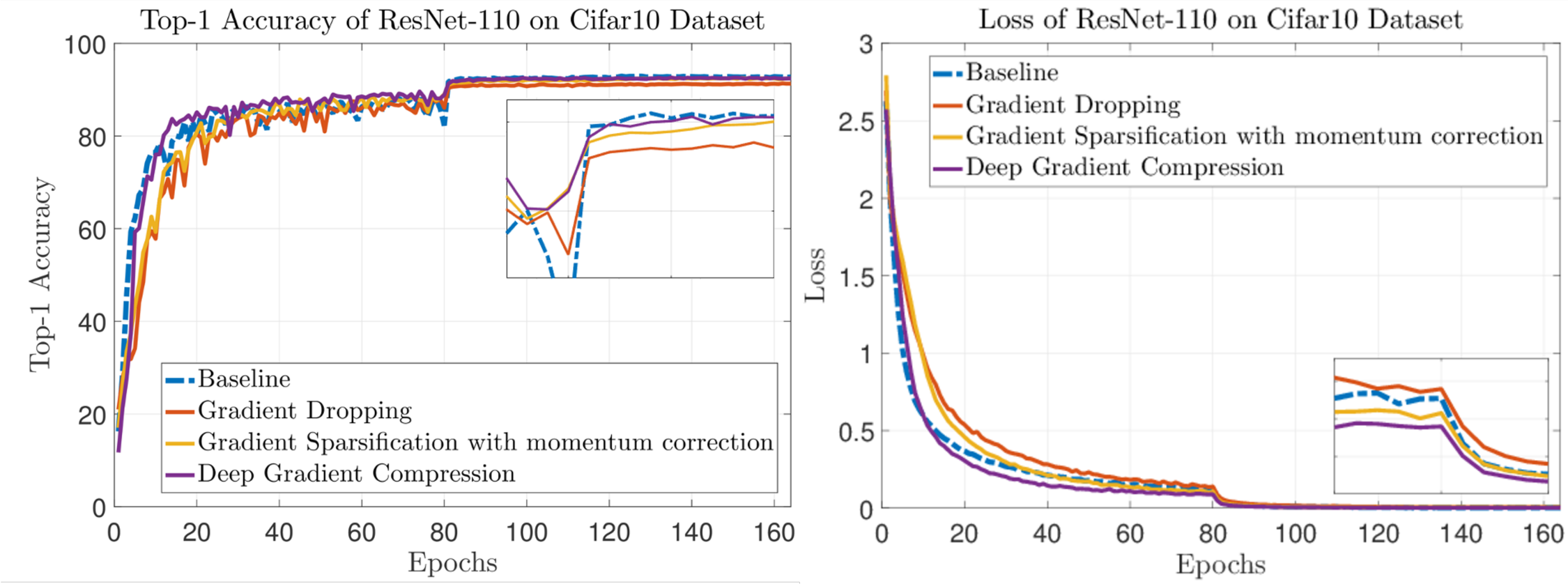
# GPUs in total	Top 1 Accuracy of ResNet-110 on Cifar10		
8	Baseline	92.92	0
	Naive Gradient Pruning	Not Converge	
	+ Local Gradient Accumulation	91.36	-1.56
	+ Local Gradient Accumulation + Momentum Correction	92.56	-0.36
	+ Local Gradient Accumulation + Warm Up Training	91.89	-1.03
	Deep Gradient Compression (Combine All Techniques)	93.28	+0.37

# GPUs in total	Word Error Rate (WER) of 5-Layer LSTM on AN4		
4	Baseline	10.76	0
	Naive Gradient Pruning	Not Converge	
	+ Local Gradient Accumulation	14.08	3.32
	+ Local Gradient Accumulation + Momentum Correction & Local Gradient Clipping	12.92	2.16
	Deep Gradient Compression (Combine All Techniques)	10.37	-0.39

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Cifar10: 99.9% gradient sparsity



Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

ImageNet: 277x-597x gradient compression

AlexNet trained on ImageNet Dataset

Training Method	Top-1	Top-5	Gradient Size	Compression Ratio
Baseline	58.17%	80.19%	232.56 MB	1 ×
DGC	58.20% (+0.03%)	80.20% (+0.01%)	0.39 MB	597 ×

Why not 1000x?
- Index overhead
- Biases are not pruned

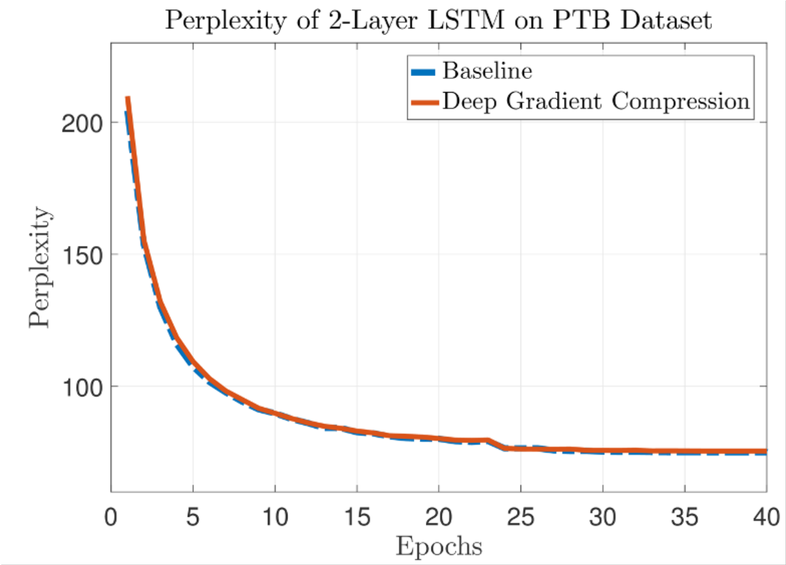
VGG trained on ImageNet Dataset

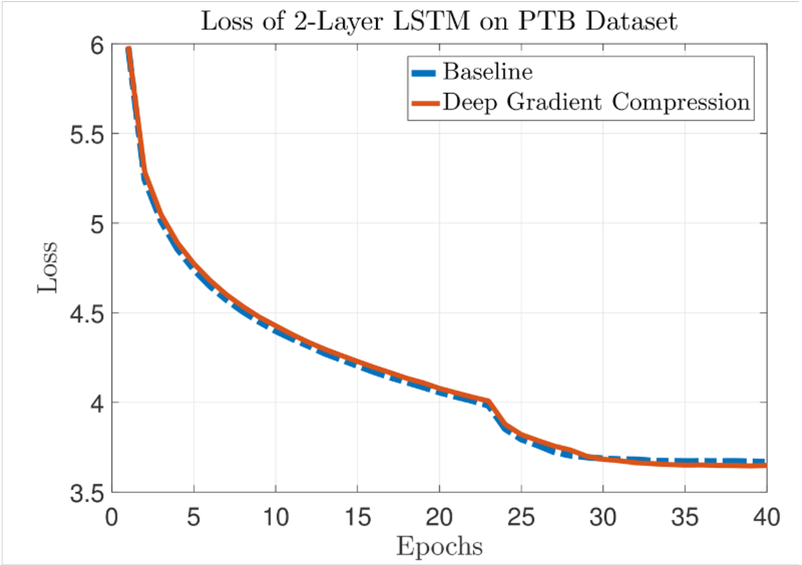
Training Method	Top-1	Top-5	Gradient Size	Compression Ratio
Baseline	75.96%	92.91%	97.49 MB	1 ×
DGC	76.15% (+0.19%)	92.97% (+0.06%)	0.35 MB	277 ×

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Language modeling: 462x gradient compression



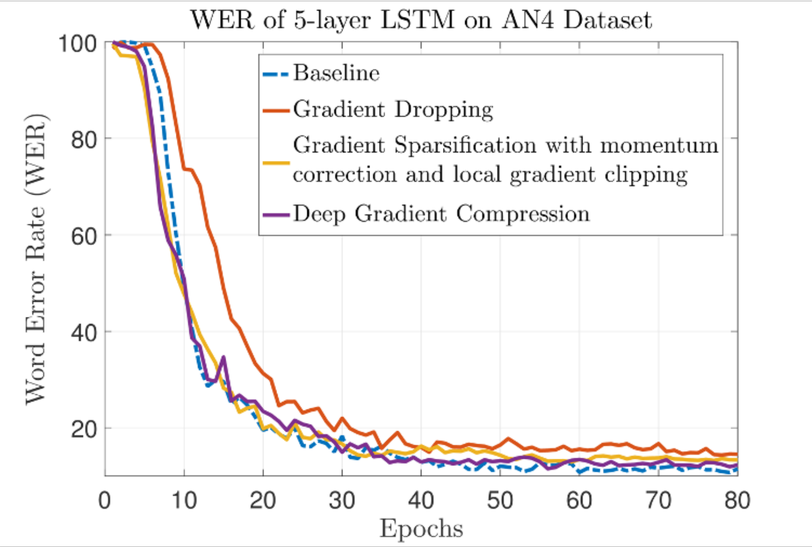


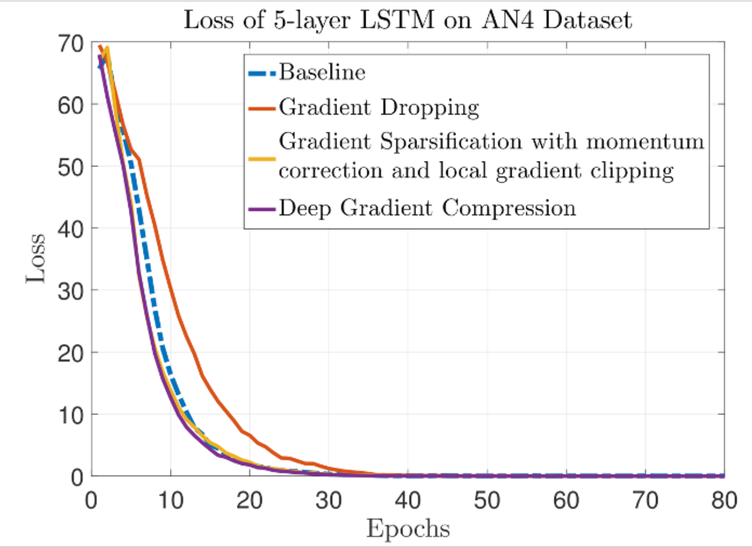
Training Method	Perplexity	Gradient Size	Compression Ratio
Baseline	72.30	194.68 MB	1 ×
Deep Gradient Compression	72.24 (-0.06)	0.42 MB	462 ×

Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Speech recognition: 608x gradient compression



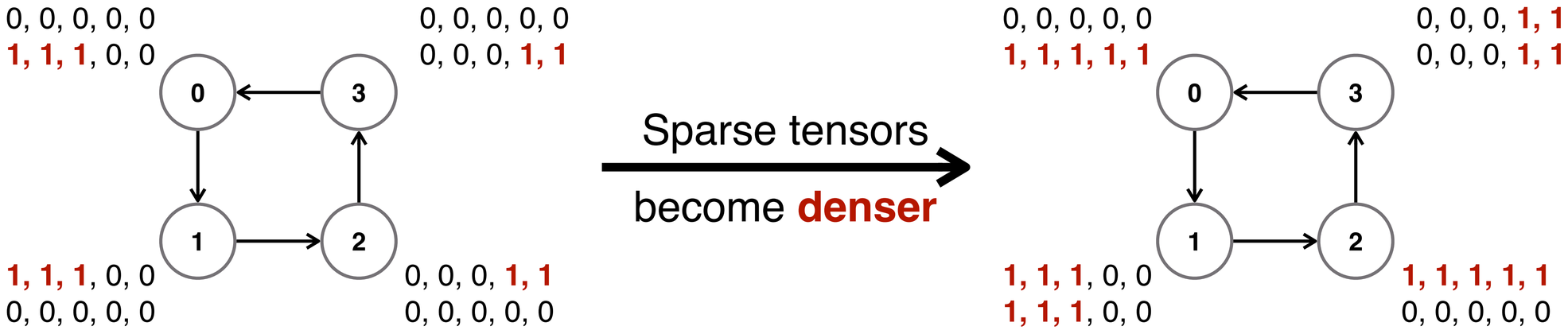


Training Method	Word Error Rate (WER) Test-clean	Test-other	Gradient Size	Compression Ratio
Baseline	9.45%	27.07%	488.08 MB	1 ×
Deep Gradient Compression	9.06% (-0.39)	27.04% (-0.03%)	0.74 MB	608 ×

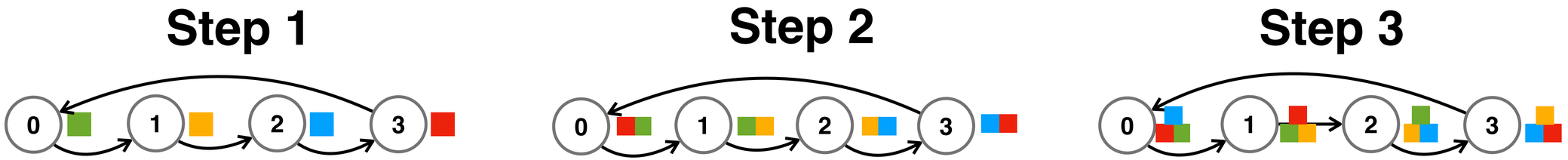
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]

Deep Gradient Compression

Problem: sparse gradients gets denser during all-reduce



AllReduce process



Solution:

1. same sparsity pattern, coarse grained sparsity [arXiv:1902.06855]
2. possibly prune in the middle of ring all-reduce

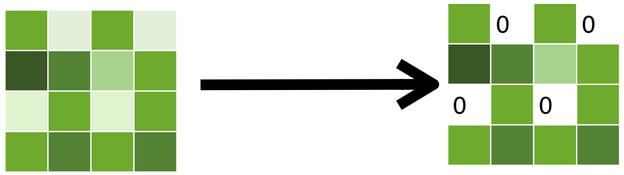
Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]
Optimizing Network Performance for Distributed DNN Training on GPU Clusters: ImageNet/AlexNet Training in 1.5 Minutes [Sun et al 2019]

PowerSGD: Low-Rank Gradient Compression

Motivation: Address the irregular sparse pattern in gradient compression, prevent gradients from getting denser

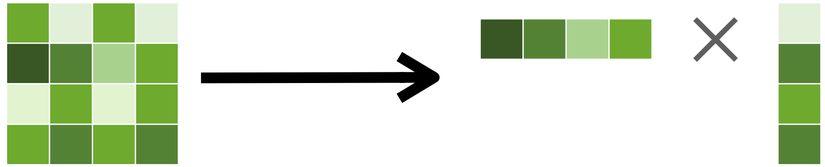
Method: Instead using fine-grained pruning, adapt low-rank factorization instead.

Deep Gradient Compression [Lin 2018]



The sparse pattern can be **different** on different servers.

PowerSGD [Vogels 2019]



The low rank matrix dimension are **same** across servers.

PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization [Vogels et al 2019]

PowerSGD: Low-Rank Gradient Compression

Speed Evaluation

There is an linear speedup when #num workers increases when using PowerSGD

PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization [Vogels et al 2019]

Comparison of Gradient Pruning Methods

- Gradient Sparsification
  - Local gradient accumulation
  - Low sparsity ratio

- Deep Gradient Compression
  - Momentum correction, gradient clipping and warmup training.
  - Higher sparsity ratio (**99.9%**) while keeping the accuracy

- PowerSGD
  - Choose low-rank instead of fine-grained pruning

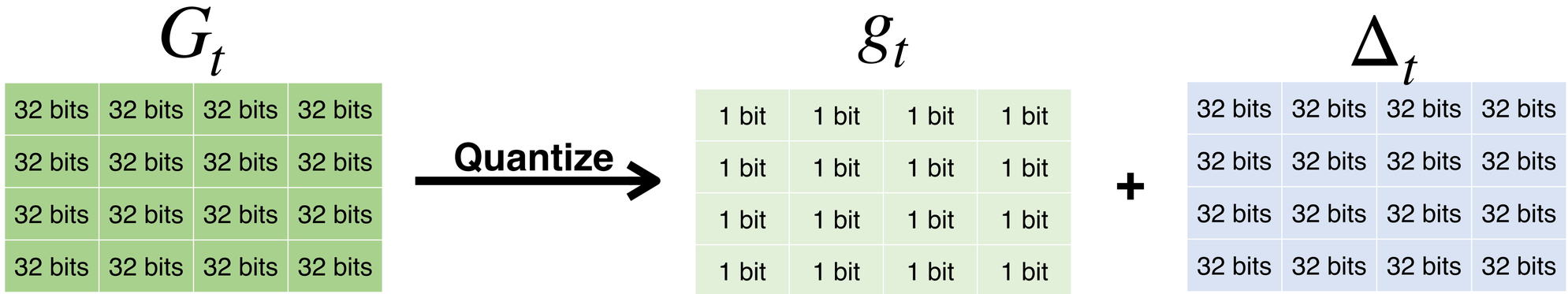
Lecture Plan

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. Understand the bandwidth and latency bottleneck of distributed training
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. **Gradient Quantization: 1-Bit SGD, TernGrad**
4. Delayed gradient update: overcome the latency bottleneck

Gradient Quantization

1 bit SGD

- With column-wise scaling factor
- Quantization error accumulation

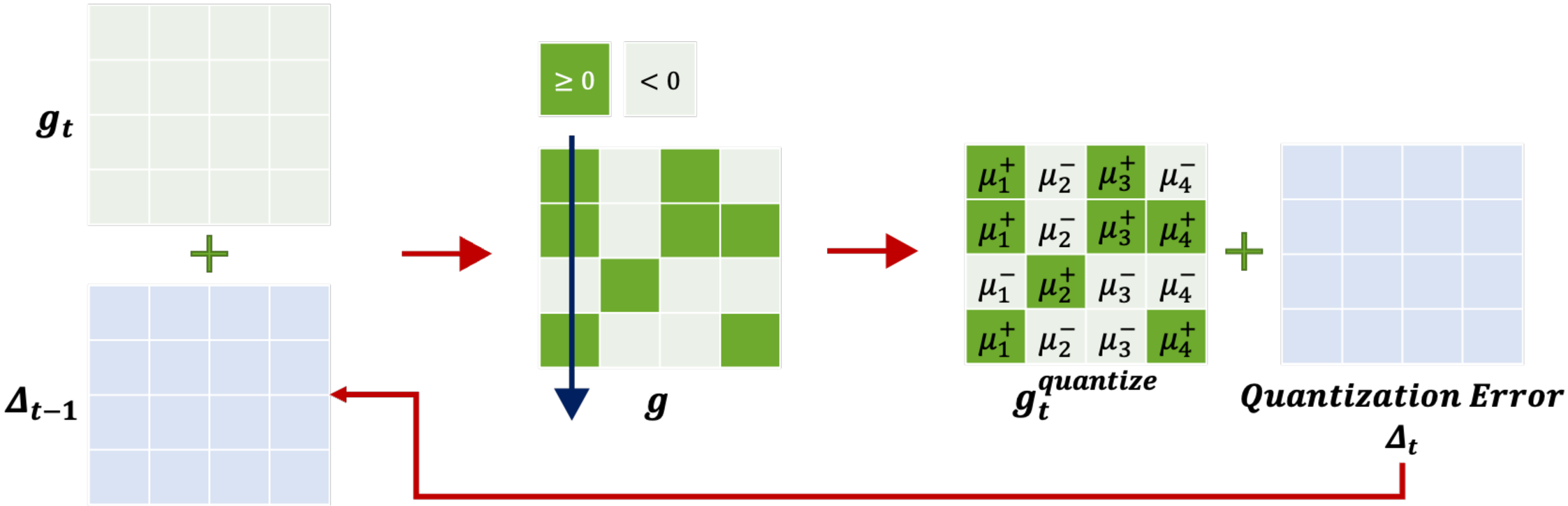


1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs [Frank 2014]

Gradient Quantization

1 bit SGD

- With column-wise scaling factor
- Quantization error accumulation

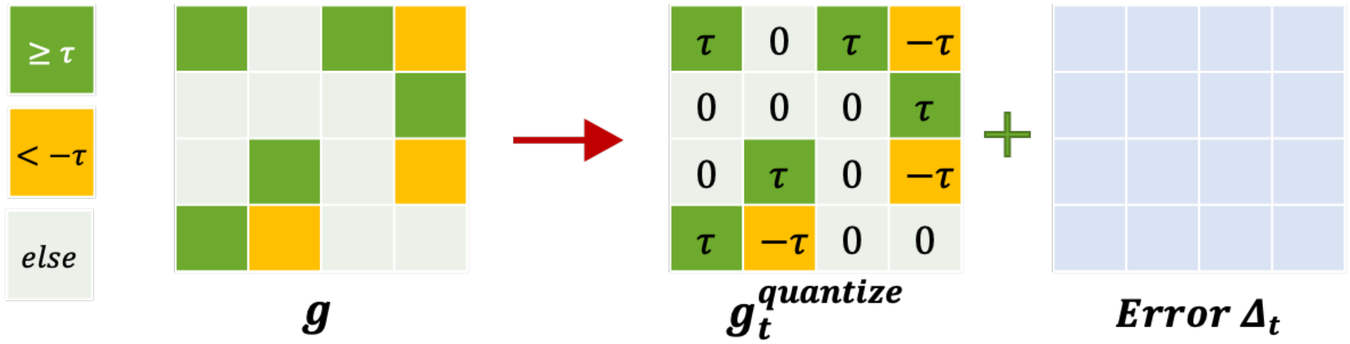


1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs [Frank 2014]

Gradient Quantization

Threshold Quantization

- τ – Threshold: τ is both threshold and reconstruction value, chosen in advance
- Quantization error accumulation
- Need to empirically choose τ

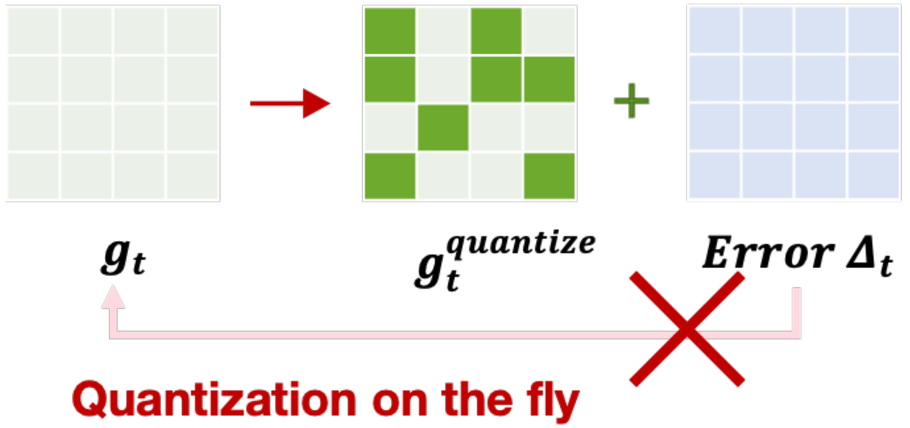


Scalable distributed DNN training using commodity GPU cloud computing [Nikko 2015]

Gradient Quantization

TernGrad

- Quantize $\frac{g_i}{\max(\mathbf{g})}$ to 0, 1, -1 with probability $\frac{|g_i|}{\max(\mathbf{g})}$, s.t. $\mathbb{E}\left(\text{Quantize}\left(g_i\right)\right) = g_i$
- No quantization error accumulation, i.e., on the fly



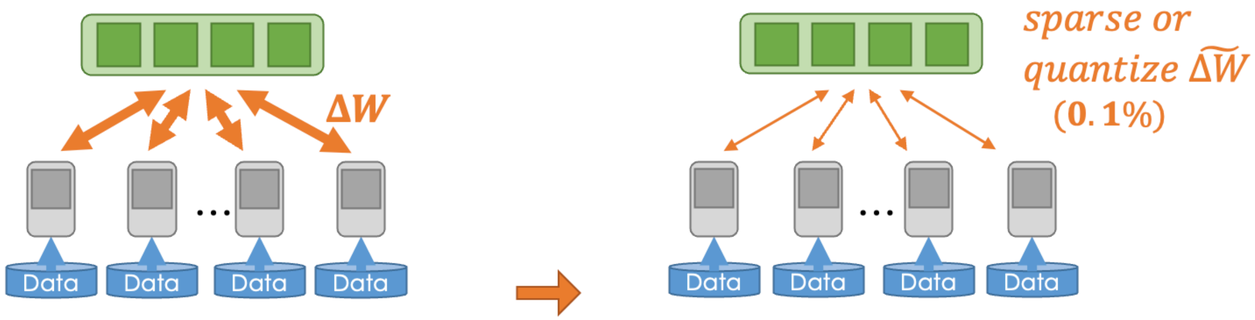
TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning

Lecture Plan

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. Understand the bandwidth and latency bottleneck of distributed training
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. **Delayed gradient update: overcome the latency bottleneck**

Bandwidth vs. Latency

Bandwidth is Easy to Improve, Latency is Hard.



Bandwidth can be always improved by

- **Gradient compression and quantization**

Gradient Pruning

- Sparse Communication
- Deep Gradient Compression
- PowerSGD

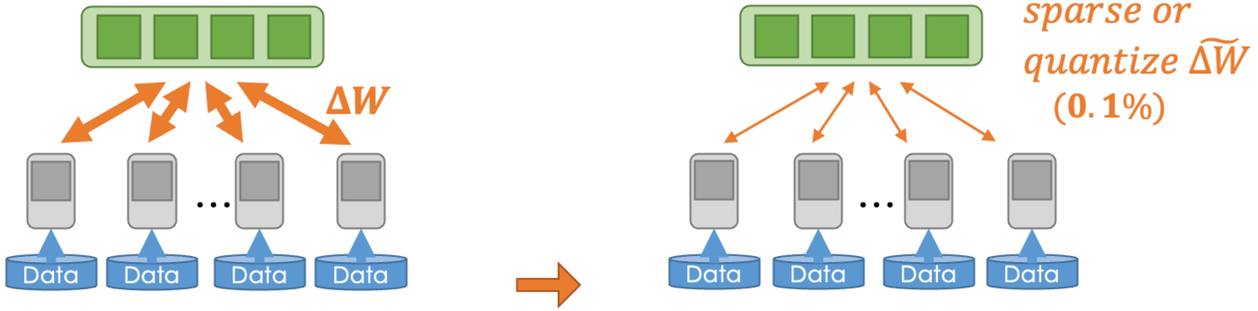
Gradient Quantization

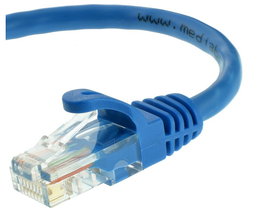
- 1-Bit SGD
- TernGrad

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Bandwidth vs. Latency

Bandwidth is Easy to Improve, Latency is Hard.





Ethernet / Home Router
100Mbps ~ 1Gbps

Fiber / Server Switch
1Gbps ~ 25Gbps

Mellanox® cables / Inifiband
20Gbps ~ 400Gbps

Bandwidth can be always improved by

- Gradient compression and quantization
- **Hardware upgrade**

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Bandwidth vs. Latency

Bandwidth is Easy to Improve, Latency is Hard.

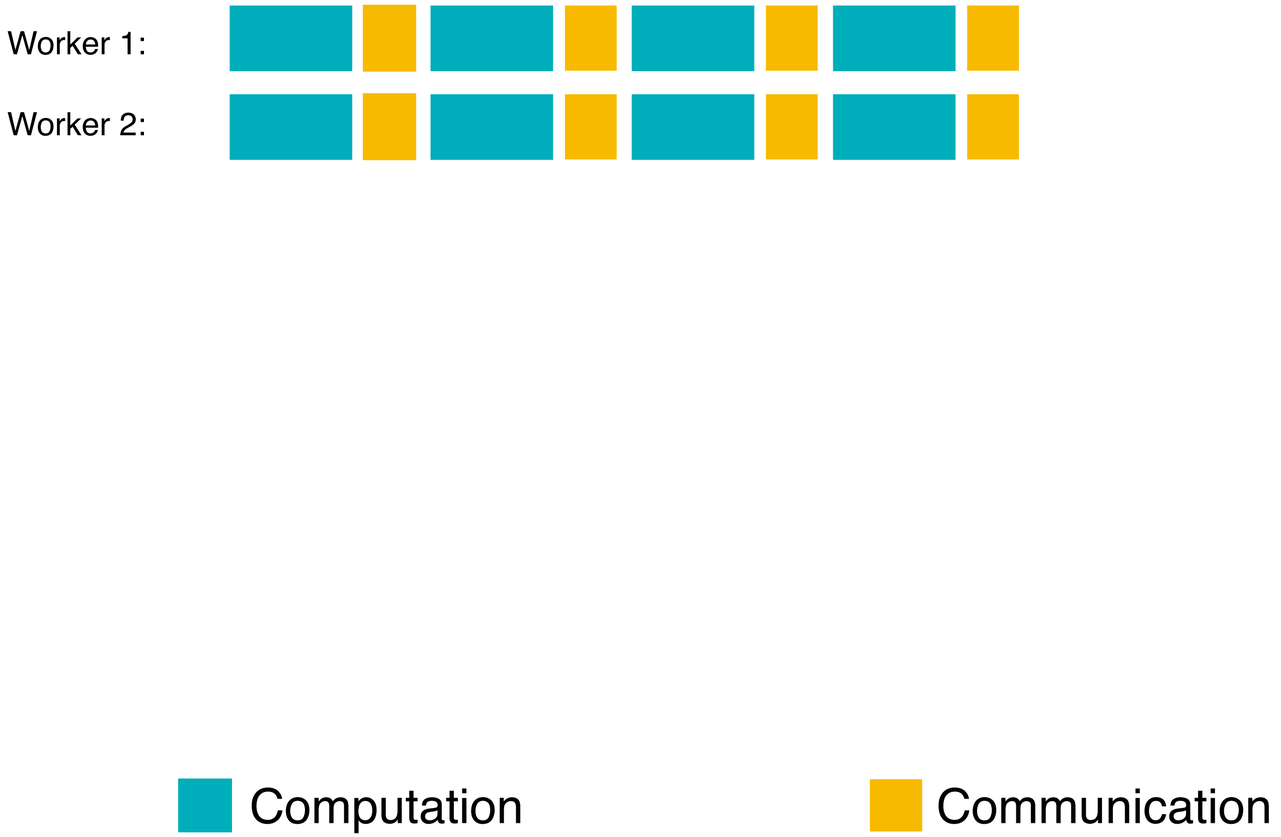
Latency **is hard to improve** because

- Physical limits: traveling from Shanghai to Boston at the speed of light still takes 162ms.
- Signal congestion: Urban office and home creates a lot of signal contention.

[1] https://www.airmilescalculator.com/distance/bos-to-pvg/

Conventional Algorithms Suffer from High Latency

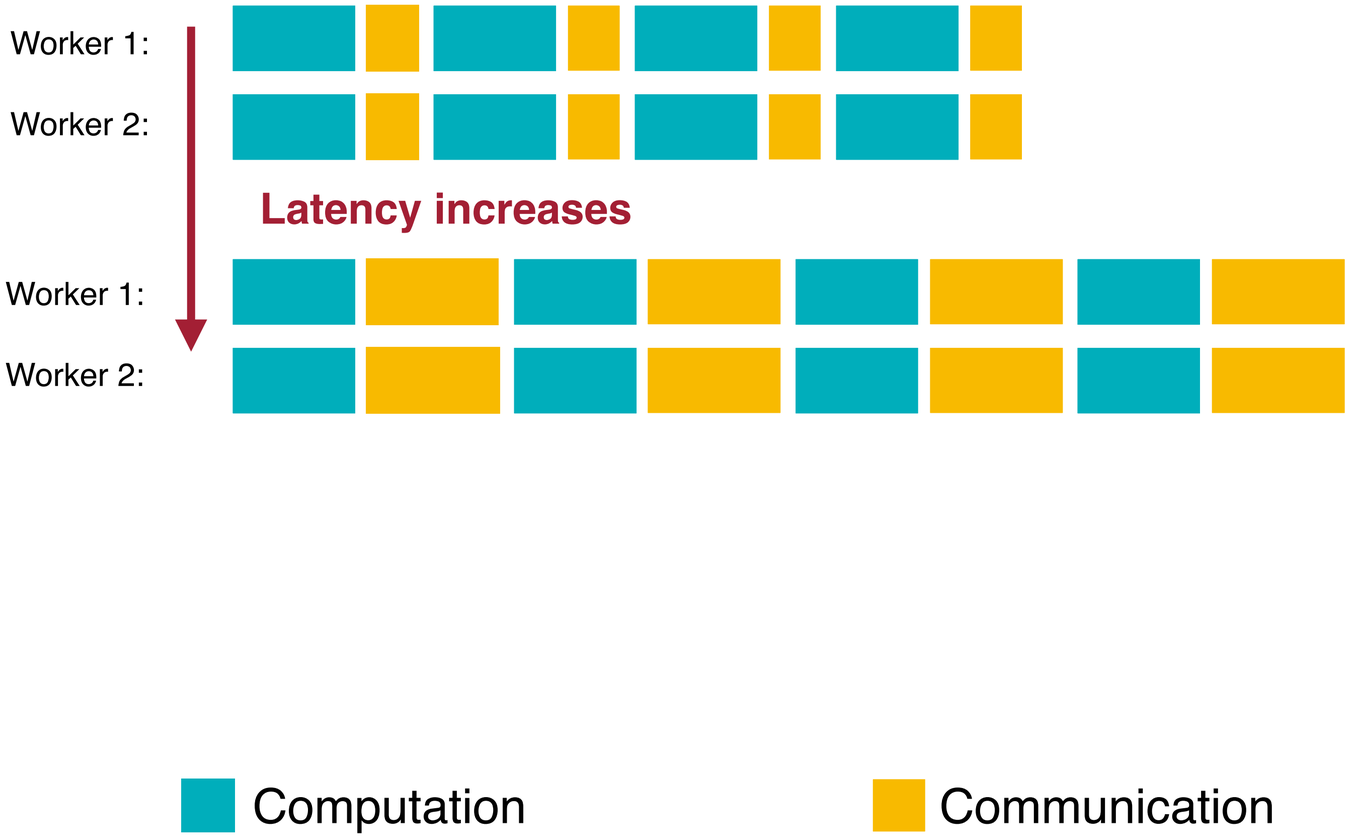
Vanilla Distributed Synchronous SGD



Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Conventional Algorithms Suffer from High Latency

Vanilla Distributed Synchronous SGD



Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Conventional Algorithms Suffer from High Latency

Vanilla Distributed Synchronous SGD

Local updates and communication are performed sequentially.
Worker **has to wait for the transmission to finish** before next step.

Computation Communication

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Conventional Algorithms Suffer from High Latency

Vanilla Distributed Synchronous SGD

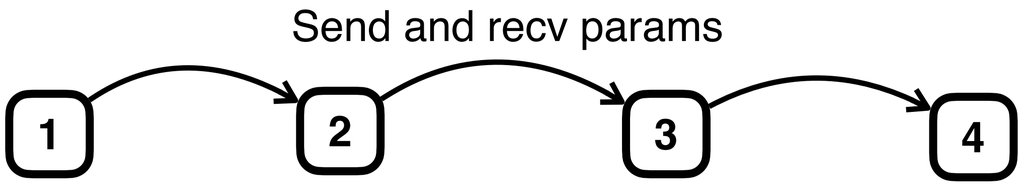
How to improve training throughput under high latency?

Can we allow stale gradients? So we can overlap communication with computation

analogy: use late days to submit your homework (gradient)

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Delayed Gradient Averaging

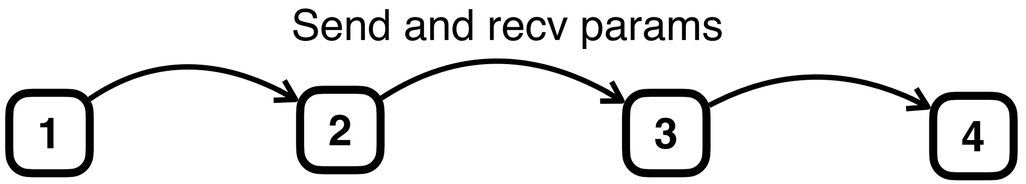


(Distributed SGD)

Without delay: all the local machines are blocked to wait for the synchronization to finish

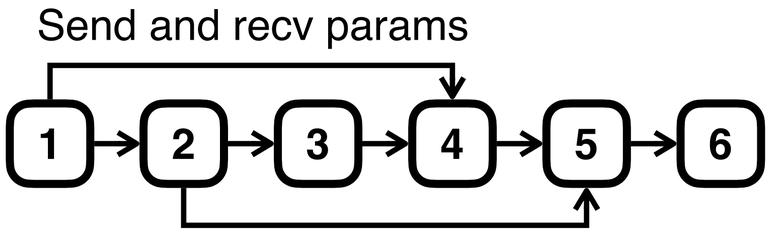
Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Delayed Gradient Averaging



(Distributed SGD)

Without delay: all the local machines are blocked to wait for the synchronization to finish



(Delayed Gradient Averaging)

With delay: Worker keep performing local updates while the parameters are in transmission.

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Delayed Gradient Averaging

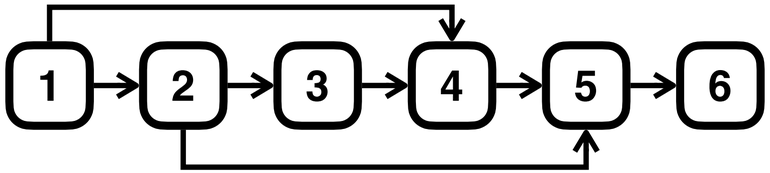
```
G = {}
for iter in 1...max_iters:
 g = grad(net, data)
 send(g, id=iter)
 G[iter] = g
 avg_g = recv(id=iter - D)
 W = W - lr * (g - G[iter-D] + avg_g)
```

Delay *D* steps

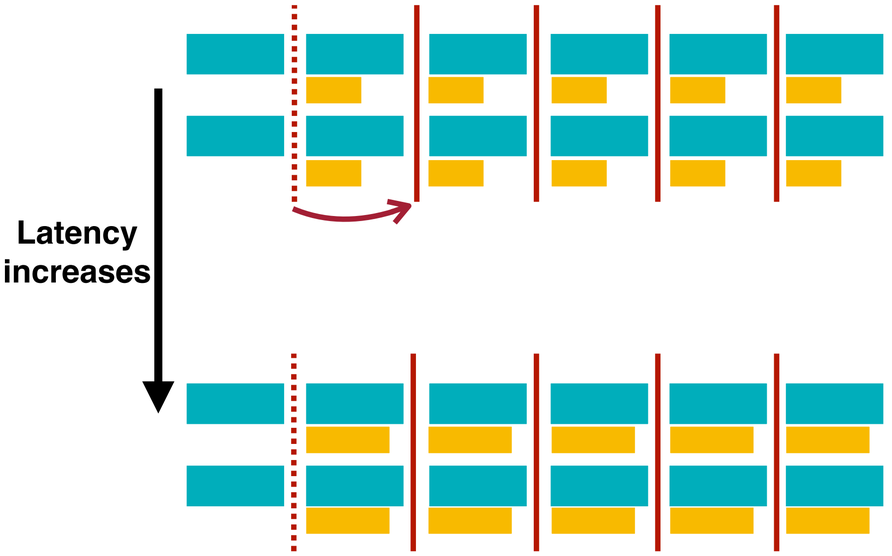
Grad from *current node*

Grad averaged from *other nodes*

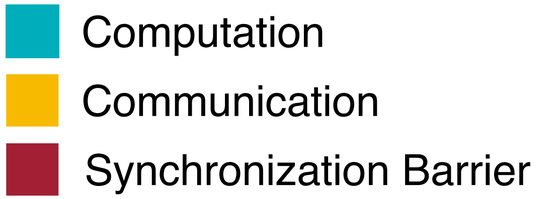
Send and recv params



Communication is covered by computation.



Total training time is not affected.



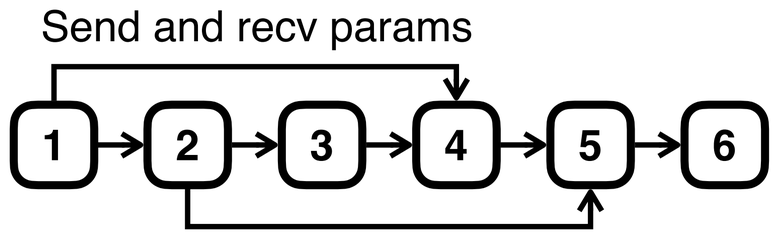
Key idea: delay the receiving timing of i^{th} gradient to a later iteration $(i+D)^{th}$

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Delayed Gradient Averaging

Deal with Staleness

- At 4^{th} iteration, the averaged gradients from 1^{st} iteration arrives (example on the right)



- How to apply this to optimizers?

- Directly apply

$$w_{(i,j)} = w_{(i,j)} - \eta \overline{\nabla w_{(i-D)}}$$

will hurt the model performance.

ResNet18, CIFAR10

	w/o correction	w/ gradient correction
D=5	88.7	89.2
D=10	86.9	89.3
D=15	85.5	89.0
D=20	84.2	88.7

- Apply with correction terms:

$$w_{(i,j)} = w_{(i,j)} - \eta(\nabla w_{(i,j)} - \nabla w_{(i-D,j)} + \overline{\nabla w_{(i-D)}})$$

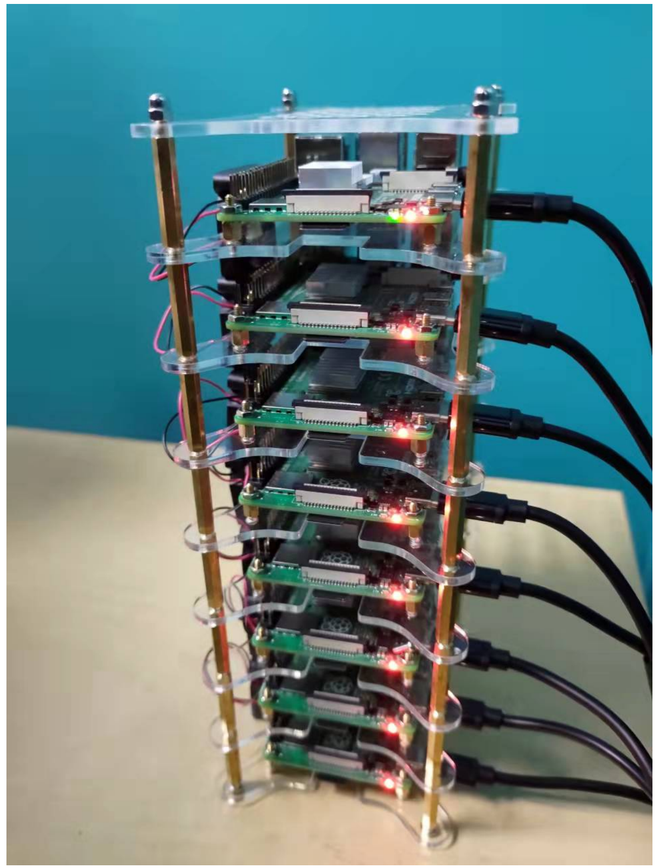
Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

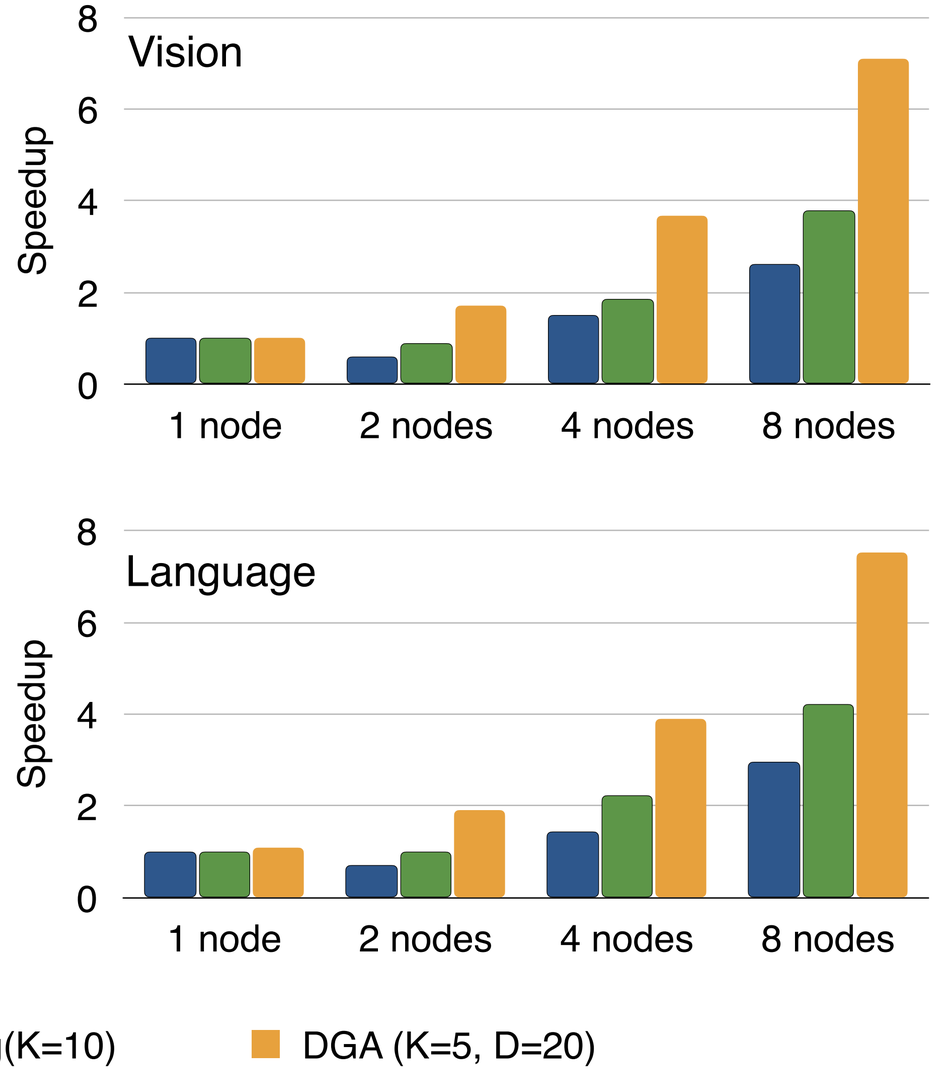
DGA Accuracy Evaluation

Datasets	Partition	FedAvg (K=5)		FedAvg (K=10)		FedAvg (K=20)		DGA (K=5, D=20)	
		Acc	Speedup	Acc	Speedup	Acc	Speedup	Acc	Speedup
CIFAR	i.i.d	88.7	1×	88.5	1.51×	88.1	2.05×	88.6	3.16×
	non-i.i.d	48.2		47.2		43.9		48.0	
ImageNet	i.i.d	76.6	1×	76.5	1.43×	76.2	1.81×	76.4	2.55×
	non-i.i.d	55.4		52.5		48.6		54.9	
Shakespeare	i.i.d	47.6	1×	47.3	1.66×	47.4	2.51×	47.1	4.07×
	non-i.i.d	36.9		34.3		30.1		36.3	

Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Real-world Benchmark





Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]

Summary of Today's Lecture

1. Hybrid (mixed) parallelism and how to auto-parallelize
2. Understand the bandwidth and latency bottleneck of distributed training
3. Gradient compression: overcome the bandwidth bottleneck
    1. Gradient Pruning: Sparse Communication, Deep Gradient Compression
    2. Gradient Quantization: 1-Bit SGD, TernGrad
4. Delayed gradient update: overcome the latency bottleneck

References

- Sparse communication for distributed gradient descent [Alham Fikri et al 2017]
- Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning [Zheng et al. 2022]
- Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training [Lin et al 2017]
- PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization [Vogels et al 2019]
- 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs [Frank 2014]
- Scalable distributed DNN training using commodity GPU cloud computing [Nikko 2015]
- QSGD: Randomized quantization for communication-optimal stochastic gradient descent [Dan et al 2016]
- Delayed Gradient Averaging: Tolerate the Communication Latency in Federated Learning. [Zhu 2021]