



Distributed Machine Learning (Part II)

Kun Yuan

Center for Machine Learning Research @ Peking University

MLSS, Okinawa

Mar. 6, 2024

Methodologies to save communication

- Global average incurs $O(n)$ comm. overhead; proportional to network size n

[Decentralized communication]

- Each node interacts with the server at every iteration; proportional to iteration numbers

[Lazy communication]

- Each node sends a full model (or gradient) to the server; proportional to dimension d

[Compressed communication]

PART 04



Lazy Communication

Distributed machine learning

A network of n nodes (devices such as GPUs) collaborate to solve the problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x), \quad \text{where } f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F(x; \xi_i).$$

- Each component $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is local and private to node i
- Random variable ξ_i denotes local data that follows distribution D_i
- Each local distribution D_i is different; data heterogeneity exists

Parallel SGD (PSGD)

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x), \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F(x; \xi_i).$$

PSGD

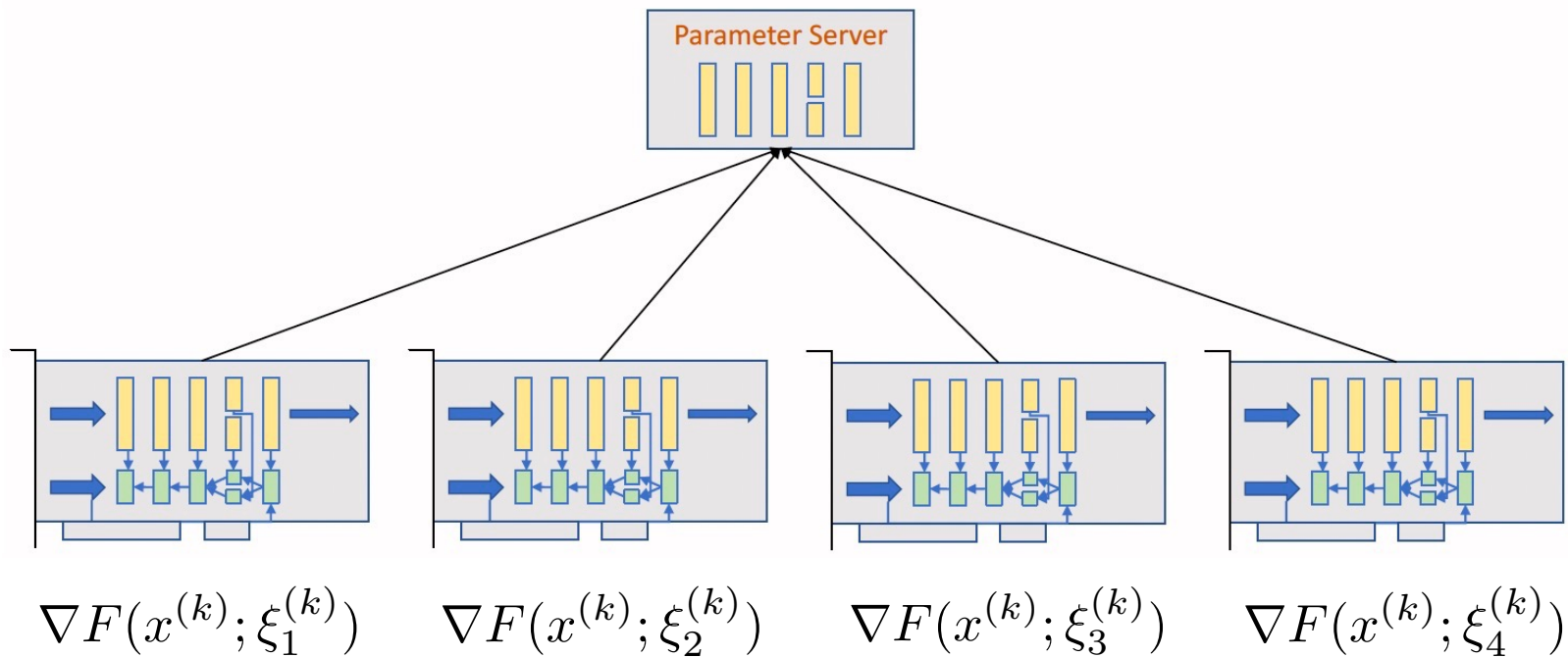
$$g_i^{(k)} = \nabla F(x^{(k)}; \xi_i^{(k)}) \quad (\text{Local compt.})$$

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^n g_i^{(k)} \quad (\text{Global comm.})$$

- Each node i samples data $\xi_i^{(k)}$ and computes gradient $\nabla F(x^{(k)}; \xi_i^{(k)})$
- All nodes synchronize (i.e. globally average) to update model x per iteration

Parallel SGD (PSGD): illustration

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^n \nabla F(x^{(k)}; \xi_i^{(k)})$$



Assumption [PSGD assumption]

- (1) Each $f_i(x)$ is L -smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y ;
- (2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

Theorem [PSGD convergence]

Under the above assumptions and with proper γ , PSGD converges as follows

$$\frac{1}{T} \sum_{k=1}^T \mathbb{E}\|\nabla f(x^{(k)})\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nT}} + \frac{1}{T}\right)$$

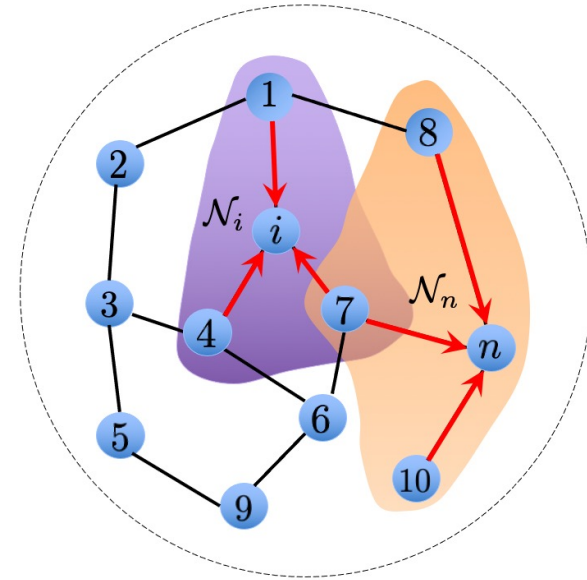
We consider more refined convergence rate to illustrate the benefits of local updates later

Decentralized SGD (DSGD)

- Decentralized SGD saves communication by **replacing global average to partial average**

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad (\text{Local update})$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} \quad (\text{Partial averaging})$$



- DSGD = local SGD update + partial averaging [1, 2]
- \mathcal{N}_i is the set of neighbors at node i ; w_{ij} scales information from j to i
- Incurs $O(d_{\max})$ comm. overhead per iteration where $d_{\max} = \max_i \{|\mathcal{N}_i|\}$ is the graph maximum degree

[1] C. G. Lopes and A. H. Sayed, "Diffusion least-mean-squares over adaptive networks", ICASSP, 2007

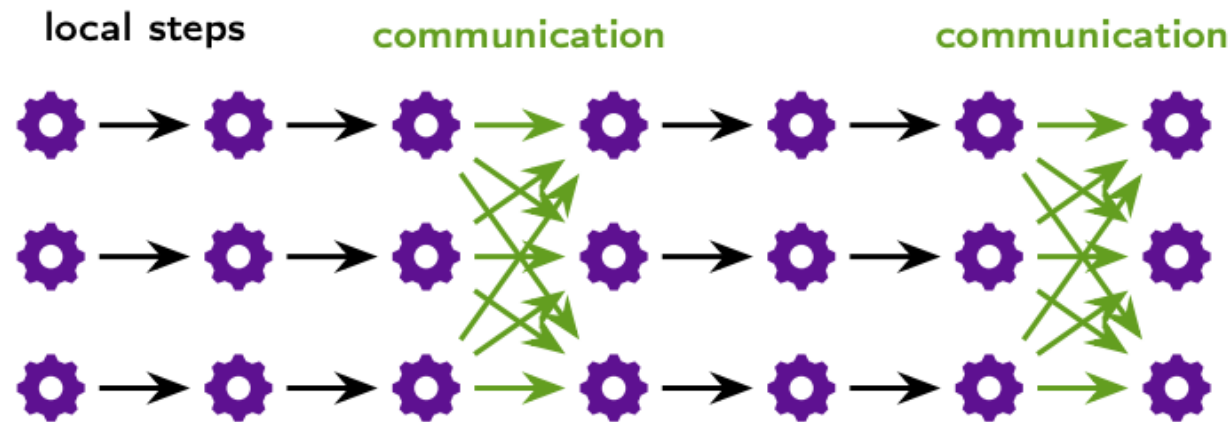
[2] A. Nedich and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization", IEEE TAC, 2009

Local SGD

- Local SGD saves communication by **reducing communication frequency**

$$x_i^{k+1} = \begin{cases} x_i^k - \gamma \nabla F(x_i^k; \xi_i^k) & \text{if } \text{mod}(k, \tau) \neq 0 \\ \frac{1}{n} \sum_{i=1}^n x_i^k & \text{if } \text{mod}(k, \tau) = 0 \end{cases}$$

- Communicate once every τ iterations



Local SGD is also known as FedAvg

- Local SGD can be rewritten in a different format

For $r = 0, 1, \dots, R - 1$ **(R comm. rounds)**

$$x_i^{(r,0)} = x^{(r)}$$

For $k = 0, 1, \dots, K - 1$ **(local update)**

$$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n x_i^{(r,K)} \quad \textbf{(global comm.)}$$

- Local SGD is also known as **FedAvg** [1,2]

[1] J. Konečný, H. B. McMahan, D. Ramage, P. Richtárik, “Federated Optimization: Distributed Machine Learning for On-Device Intelligence”, 2016

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, AISTATS 2017

Assumption [Local SGD assumption]

(1) Each $f_i(x)$ is L -smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y ;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

(3) Gradient dissimilarity can be upper bounded, i.e.,

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq b^2$$

Same as DSGD assumptions

Theorem [Local SGD convergence]

Under the above assumptions, by setting local learning rate and global learning rate properly, it holds that [1]

$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{b^{2/3}}{R^{2/3}} + \frac{1}{R} \right)$$

- The number of local updates K **improves** the dominant term in the convergence rate
- Parallel SGD with R communication rounds will converge at rate

$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nR}} + \frac{1}{R} \right)$$

- Local SGD converges faster than PSGD with the same communication budgets

Local SGD v.s. parallel SGD

- It is intuitive that local SGD converges faster than PSGD; **local SGD runs K times more updates**

Local SGD

For $r = 0, 1, \dots, R - 1$ **(R comm. rounds)**

$$x_i^{(r,0)} = x^{(r)}$$

For $k = 0, 1, \dots, K - 1$ **(local update)**

$$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n x_i^{(r,K)} \quad \textbf{(global comm.)}$$

Parallel SGD

For $r = 0, 1, \dots, R - 1$ **(R comm. rounds)**

$$\phi_i^{(r+1)} = x^{(r)} - \gamma \nabla F(x^{(r)}, \xi_i^{(r)})$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n \phi_i^{(r+1)} \quad \textbf{(global comm.)}$$

Local SGD v.s. mini-batch parallel SGD

- It is more fair to compare local SGD and parallel SGD with batch-size K

Local SGD

For $r = 0, 1, \dots, R - 1$ (**R comm. rounds**)

$$x_i^{(r,0)} = x^{(r)}$$

For $k = 0, 1, \dots, K - 1$ (**local update**)

$$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n x_i^{(r,K)} \quad \text{(global comm.)}$$

Mini-batch Parallel SGD

For $r = 0, 1, \dots, R - 1$ (**R comm. rounds**)

$$\phi_i^{(r+1)} = x^{(r)} - \frac{\gamma}{K} \sum_{k=1}^K \nabla F(x^{(r)}, \xi_i^{(r,k)})$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n \phi_i^{(r+1)} \quad \text{(global comm.)}$$

Local SGD v.s. mini-batch parallel SGD

- Convergence comparison

Local SGD:
$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{b^{2/3}}{R^{2/3}} + \frac{1}{R} \right)$$

Mini-batch parallel SGD:
$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{R} \right)$$

- Local SGD is worse than mini-batch parallel SGD due to the **influence of data heterogeneity**

Local SGD v.s. mini-batch parallel SGD

- In real practices, local SGD outperforms mini-batch SGD when data heterogeneity is not severe, i.e., training deep neural network in data-centers
- **Large mini-batch causes severe generalization degradation; while local SGD does not**

Local SGD yields better generalization than huge batches

| | Top-1 acc. | local gradients | communication |
|---|--------------|-----------------|---------------|
| Mini-batch SGD ($n = 16, \tau = 128$) | 92.5% | 2048 | - |
| Mini-batch SGD ($n = 16, \tau = 1024$) | 76.3% | 16384 | $\div 8$ |
| Local-SGD ($n = 16, \tau = 8 \times 128$) | 92.0% | 16384 | $\div 8$ |

ResNet-20 on Cifar-10

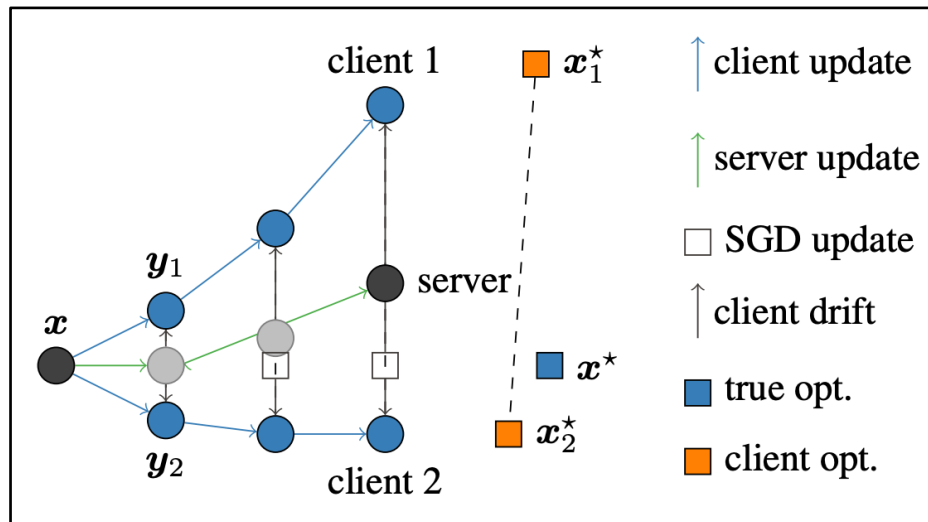
[1] S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

[2] T. Lin et. al., "Don't use large mini-batches, use local SGD", ICLR 2020

Data heterogeneity significantly influences local SGD

- In federated learning, local data distributions vary drastically; strong data heterogeneity exists
- Local SGD gets significantly degraded when strong data heterogeneity exists

$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{b^{2/3}}{R^{2/3}} + \frac{1}{R} \right)$$



Performance drops significantly

| Method | $\alpha = 0.1$ | $\alpha = 0.5$ | $\alpha = 2.5$ | homog |
|---------|----------------|----------------|----------------|----------|
| FedAvg | 45.0±0.2 | 52.9±0.1 | 54.4±0.2 | 54.9±0.4 |
| FedProx | 45.2±0.3 | 53.1±0.3 | 54.5±0.3 | 54.8±0.5 |
| MOON | 46.5±0.5 | 55.0±0.5 | 56.3±0.6 | 56.3±0.5 |

[M. Mendieta et. al., CVPR 2022]

Scaffold: an effective algorithm to remove data heterogeneity

- In local SGD, after K local updates, we have

$$x_i^{(r,K)} = x_i^{(r,0)} - \gamma g_i^{(r)}$$

where
$$g_i^{(r)} = \frac{\gamma}{K} \sum_{k=0}^{K-1} \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$$

For $r = 0, 1, \dots, R - 1$ **(R comm. rounds)**

$$x_i^{(r,0)} = x^{(r)}$$

For $k = 0, 1, \dots, K - 1$ **(local update)**

$$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$$

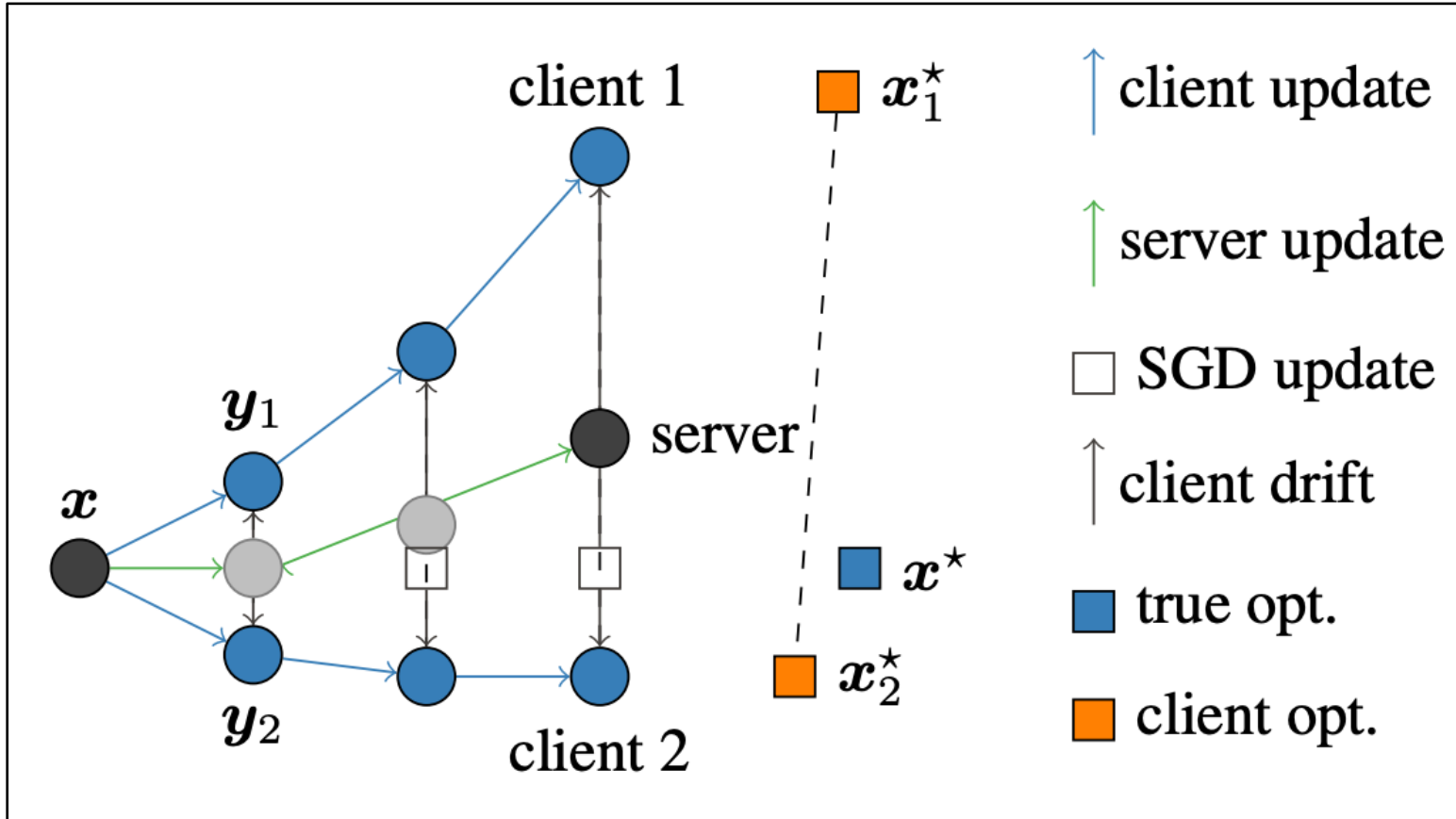
$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n x_i^{(r,K)} \quad \textbf{(global comm.)}$$

- The root reason why local SGD suffers from data heterogeneity is

$g_i^{(r)}$ estimates $\nabla f_i(x^{(r)})$ well but is far away from the globally-averaged $(1/n) \sum_{j=1}^n \nabla f_j(x^{(r)})$

- $g_i^{(r)}$ is towards the local minimum, not the global minimum

Scaffold: an effective algorithm to remove data heterogeneity



Local update move towards local solutions, not the global solution

Scaffold: an effective algorithm to remove data heterogeneity

- To correct the gradient, Scaffold introduces the following approach

$$x_i^{(r,K)} = x_i^{(r,0)} - \gamma \left(g_i^{(r)} - c_i^{(r)} + c^{(r)} \right)$$

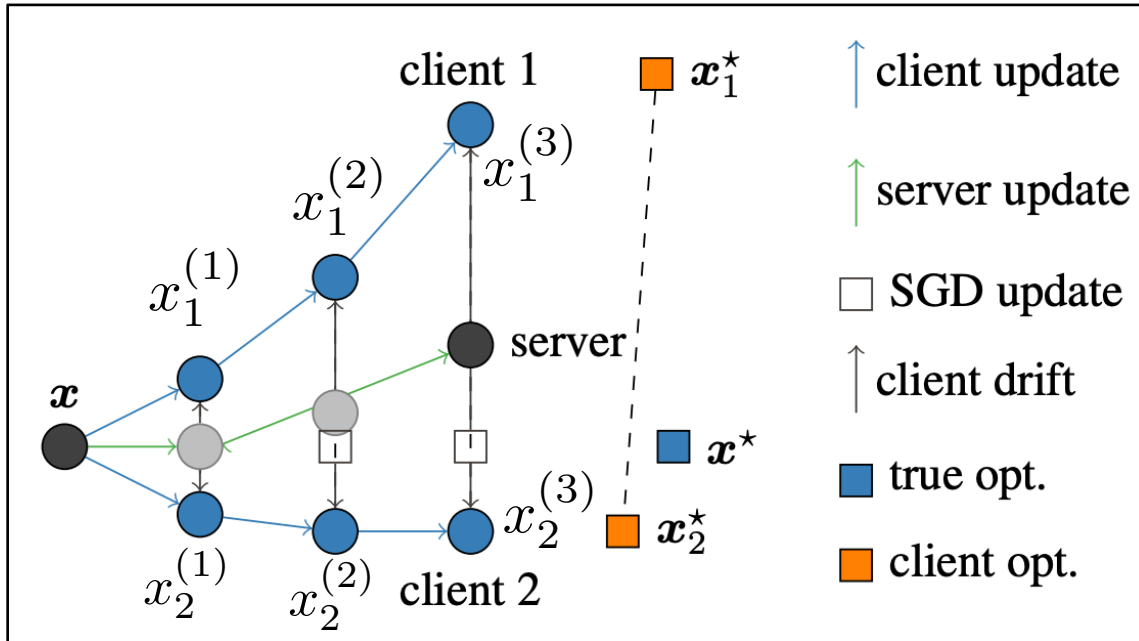
where $c_i^{(r)} = g_i^{(r-1)}$ and $c^{(r)} = (1/n) \sum_{j=1}^n c_j^{(r)}$

- Suppose $g_i^{(r)} \rightarrow \nabla f_i(x^{(r)})$, the corrected gradient $g_i^{(r)} - c_i^{(r)} + c^{(r)} \rightarrow \frac{1}{n} \sum_{i=1}^n f_i(x^{(r)})$
- Local update drives each local model towards the global minimum

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

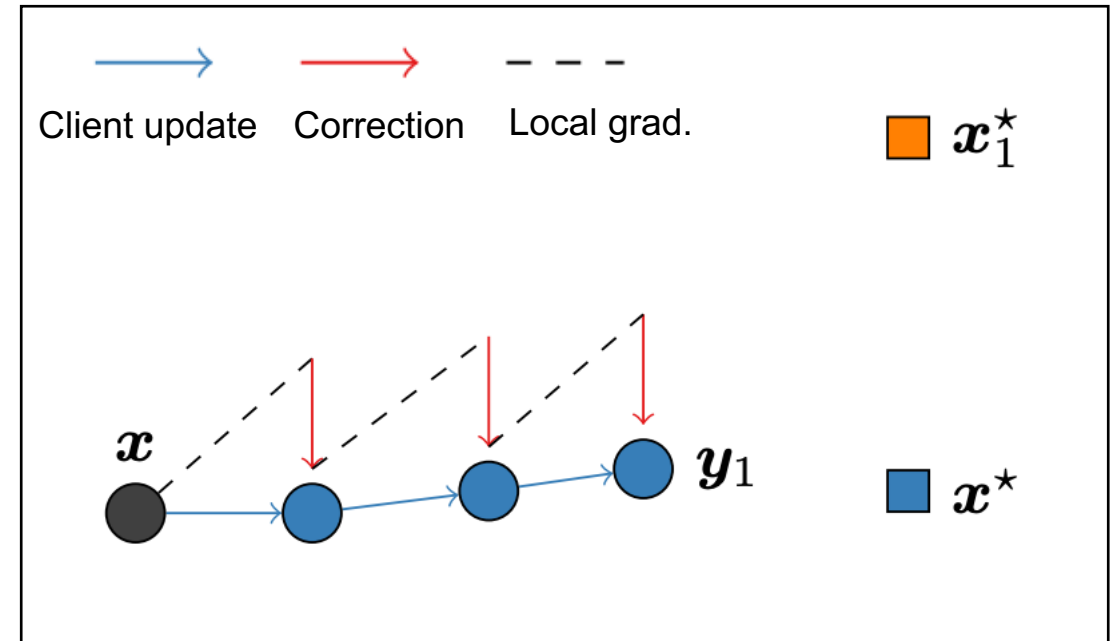
Local SGD v.s. Scaffold

Local SGD



$$x_i^* \neq x^* \implies (1/n) \sum_{i=1}^n x_i^* \neq x^*$$

Scaffold



$$x_i^* \rightarrow x^* \implies (1/n) \sum_{i=1}^n x_i^* \rightarrow x^*$$

Scaffold algorithm

For $r = 0, 1, \dots, R - 1$ **(R comm. rounds)**

$$x_i^{(r,0)} = x^{(r)} \quad c_i^{(r+1)} = 0$$

For $k = 0, 1, \dots, K - 1$ **(K local updates)**

$$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma \left(\nabla F(x_i^{(r,k)}, \xi_i^{(r,k)}) - c_i^{(r)} + c^{(r)} \right)$$

$$c_i^{(r+1)} += \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)}) / K$$

$$x^{(r+1)} = \frac{1}{n} \sum_{i=1}^n x_i^{(r,K)} \quad c^{(r+1)} = \frac{1}{n} \sum_{i=1}^n c_i^{(r)} \quad \textbf{(global comm.)}$$

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

Assumption [Scaffold assumptions]

- (1) Each $f_i(x)$ is L -smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y ;
- (2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

Theorem [Scaffold convergence]

Under the above assumptions and with proper γ , Scaffold converges as follows

$$\frac{1}{R} \sum_{r=1}^R \mathbb{E}\|\nabla f(x^{(r)})\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{R}\right)$$

Scaffold removes the influence of data heterogeneity

Convergence comparison

Local SGD:
$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{b^{2/3}}{R^{2/3}} + \frac{1}{R} \right)$$

Mini-batch parallel SGD:
$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{R} \right)$$

Scaffold:
$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{R} \right)$$

- We cannot show the superiority of Scaffold to mini-batch parallel SGD
- It is believed in the community that local updates **cannot** bring theoretical convergence benefits within the family of general non-convex and smooth problems [1]

[1] B. Woodworth et. al., "Is Local SGD Better than Minibatch SGD?", ICML 2020

Provable benefits under additional assumptions

- in the standard complexity classes, no benefit of local steps!
- however, on quadratic functions we hope to be better!
 - IID setting: (same Hessian, matching local minima)

| algorithm | rounds |
|-------------------------------------|---|
| mini-batch SGD batch size τ | $\mathcal{O}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\mu} \log \frac{1}{\epsilon}\right)$ |
| local SGD τ local steps | $\mathcal{O}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\tau\mu} \log \frac{1}{\epsilon}\right)$ |

- non-IID setting: (same Hessian, different local minima)

| | |
|--------------------------------|---|
| SCAFFOLD τ local steps | $\tilde{\mathcal{O}}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\tau\mu} \log \frac{1}{\epsilon}\right)$ |
|--------------------------------|---|

Remarkable: benefit from parallelism and serial updates!

Scaffold performs well in non-iid scenarios

Communication rounds to reach 0.5 test accuracy for logistic regression on EMNIST

| | | non-IID data | | | | IID data | |
|------------|--------|------------------------|---------|----------------|---------|--------------------------|---------|
| Algorithms | Epochs | 0% similarity (sorted) | | 10% similarity | | 100% similarity (i.i.d.) | |
| | | Num. of rounds | Speedup | Num. of rounds | Speedup | Num. of rounds | Speedup |
| SGD | 1 | 317 | (1×) | 365 | (1×) | 416 | (1×) |
| | 5 | 77 | (4.1×) | 62 | (5.9×) | 60 | (6.9×) |
| SCAFFOLD1 | 1 | 152 | (2.1×) | 20 | (18.2×) | 10 | (41.6×) |
| | 5 | 258 | (1.2×) | 74 | (4.9×) | 83 | (5×) |
| FEDAVG | 1 | 428 | (0.7×) | 34 | (10.7×) | 10 | (41.6×) |
| | 5 | | | | | | |

less rounds with drift correction

no drift correction needed

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

In federated learning

- Server cannot control user's device and data. Users join and leave the learning process at their will (that's why it is called **Federated Learning**)
- Clients are not stable; they are not available when out of power or signal

This implies that not all clients can participate in the training or learning process

Client sampling: sample $s \leq n$ clients per iteration during the training process

Does not change the algorithm and analysis too much

- Client sampling does not change the algorithm and analysis too much

Scaffold with client sampling

$$\frac{1}{R} \sum_{r=1}^R \mathbb{E} \|\nabla f(x^{(r)})\|^2 = \mathcal{O} \left(\frac{\sigma}{\sqrt{sKR}} + \frac{1}{R} \left(\frac{n}{s}\right)^{2/3} \right)$$

- When $n = s$, the above results reduce to Scaffold with full-client sampling
- Fewer sampled clients lead to slower convergence performance
- The above results hold for **uniform** client sampling; Arbitrary client sampling is less explored

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

Momentum benefits local SGD simply and provably

- Momentum is widely-used in federated learning and local SGD
- Its theoretical benefits to federated learning algorithms are not well-understood

Algorithm 1 FEDAVG-M: FEDAVG with momentum

Require: initial model x^0 and gradient estimate g^0 , local learning rate η , global learning rate γ , momentum β

for $r = 0, \dots, R - 1$ **do**

for each client $i \in \{1, \dots, N\}$ in parallel **do**

 Initialize local model $x_i^{r,0} = x^r$

for $k = 0, \dots, K - 1$ **do**

 Compute $g_i^{r,k} = \beta \nabla F(x_i^{r,k}; \xi_i^{r,k}) + (1 - \beta)g^r$

 Update local model $x_i^{r,k+1} = x_i^{r,k} - \eta g_i^{r,k}$

end for

end for

 Aggregate local updates $g^{r+1} = \frac{1}{\eta NK} \sum_{i=1}^N (x^r - x_i^{r,K})$

 Update global model $x^{r+1} = x^r - \gamma g^{r+1}$

end for

▷ $\beta = 1$ implies FEDAVG

$$\beta \rightarrow 0 \implies g_i^{(r,k)} \rightarrow g^{(r)}$$

Beta tunes how local gradient changes towards global gradient

Z. Cheng, et. al., "Momentum Benefits NON-IID Federated Learning Simply and Provably", ICLR 2024

Momentum benefits local SGD simply and provably

- For local SGD, momentum provably removes data heterogeneity; no need for Scaffold

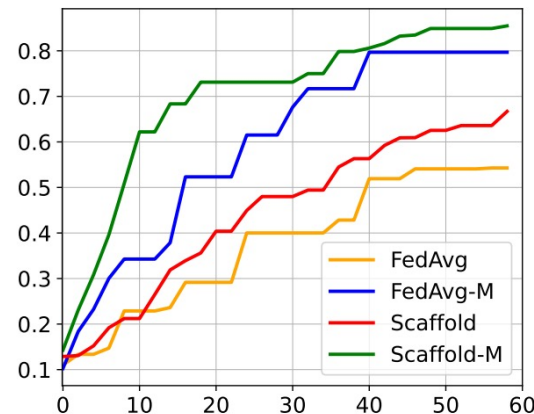
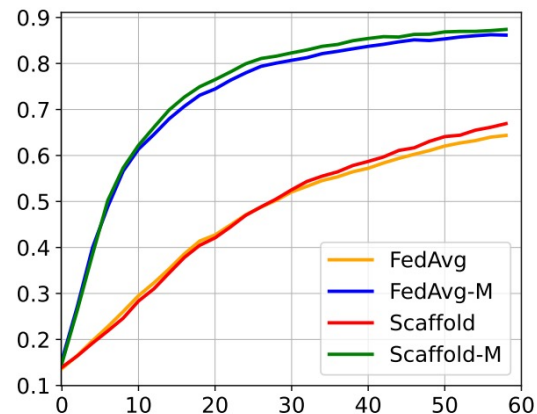
| Algorithm | Convergence Rate $\mathbb{E}[\ \nabla f(\hat{x})\ ^2] \lesssim$ | Assumptions |
|--|---|----------------------------------|
| FEDAVG | | |
| (Yu et al., 2019b) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta G}{R}\right)^{2/3} + \frac{L\Delta}{R}$ | Bounded grad. |
| (Koloskova et al., 2020) | $\left(\frac{L\Delta\sigma^2}{NR}\right)^{1/2} + \left(\frac{L\Delta K\zeta}{R}\right)^{2/3} + \frac{L\Delta K}{R}$ | Bounded hetero. |
| (Karimireddy et al., 2020b) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\zeta}{R}\right)^{2/3} + \frac{L\Delta}{R}$ | Bounded hetero. |
| (Yang et al., 2021) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \frac{L\Delta}{R}$ | Bounded hetero. ¹ |
| FEDCM ² (Xu et al., 2021b) | $\left(\frac{L\Delta(\sigma^2 + NKG^2)}{NKR}\right)^{1/2} + \left(\frac{L\Delta(\sigma/\sqrt{K} + G)}{R}\right)^{2/3}$ | Bounded grad. Bounded hetero. |
| LED (Algunaim, 2023) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\sigma}{\sqrt{KR}}\right)^{2/3} + \frac{L\Delta}{R}$ | — |
| VRL-SGD ² (Liang et al., 2019) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\sigma}{\sqrt{KR}}\right)^{2/3} + \frac{L\Delta}{R}$ | — |
| FEDAVG-M (Thm. 1) | $\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \frac{L\Delta}{R}$ No data heterogeneity | |

Z. Cheng, et. al., “Momentum Benefits NON-IID Federated Learning Simply and Provably”, ICLR 2024

Momentum benefits local SGD simply and provably

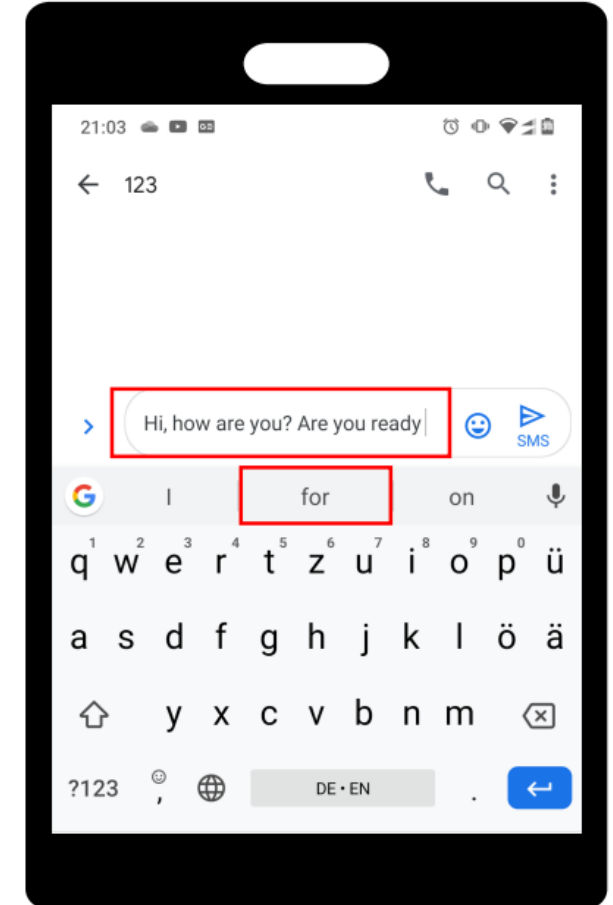
- For Scaffold, momentum improves its convergence rate
- Can achieve up to $n^{2/9}$ times speedup compared to vanilla Scaffold; state-of-the-art rate

| Algorithm | Convergence Rate $\mathbb{E}[\ \nabla f(\hat{x})\ ^2] \lesssim$ | Assumptions |
|---|---|-------------|
| SCAFFOLD (Karimireddy et al., 2020b) | $\left(\frac{L\Delta\sigma^2}{SKR}\right)^{1/2} + \frac{L\Delta}{R} \left(\frac{N}{S}\right)^{2/3}$ | — |
| SCAFFOLD-M (Thm. 3) | $\left(\frac{L\Delta\sigma^2}{SKR}\right)^{1/2} + \frac{L\Delta}{R} \left(1 + \frac{N^{2/3}}{S}\right)$ | — |



Personalized federated learning

- The above discussions target to learn a global model for all clients
- Personalization is highly preferred in federated learning, e.g., cellphone input engine
- Learn a global model for all clients cannot promote personalization
- **Shared structure** + **local head** balances between common model and personalized model

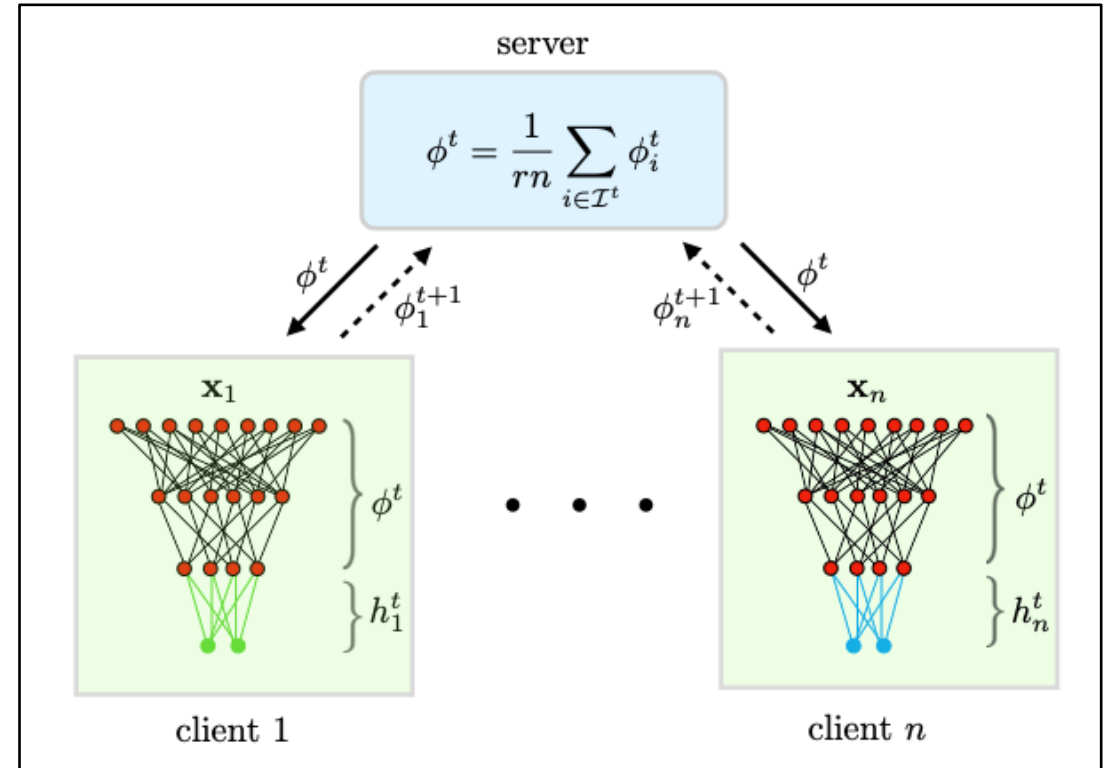


Personalized federated learning

$$\min_{u, \mathbf{v}} f(u, \mathbf{v}) := \frac{1}{n} \sum_{i=1}^n f_i(u, v_i)$$

where $f_i(u, v_i) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [F(u, v_i; \xi_i)]$.

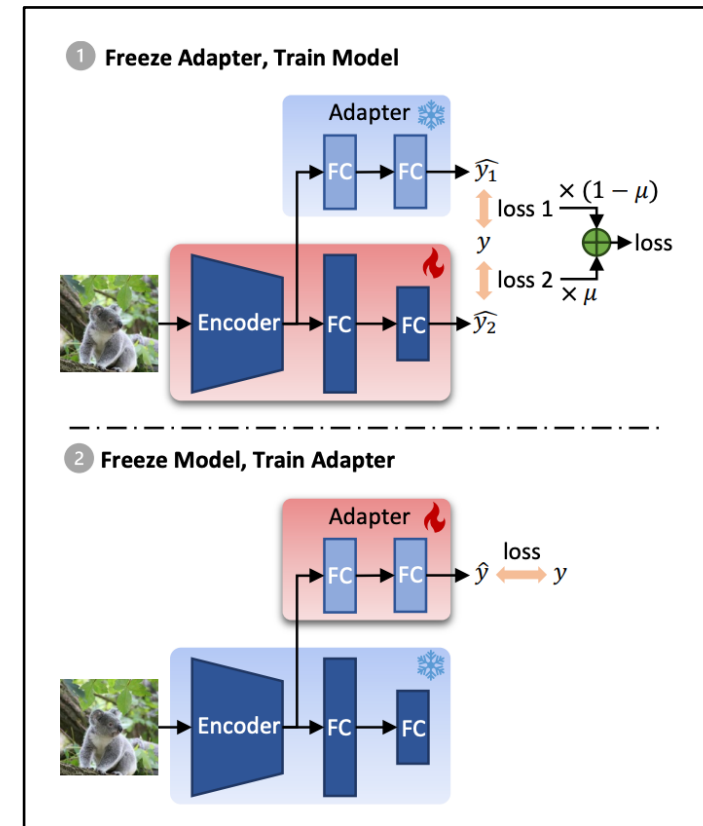
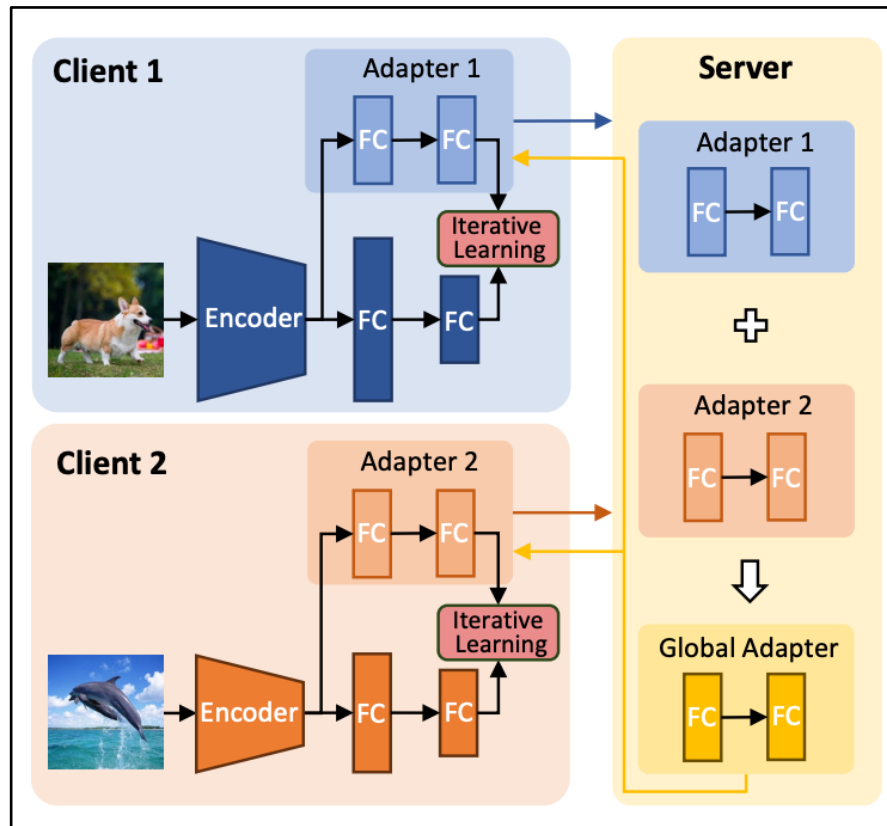
- u represents common structures
- v_i represents local models
- End-to-end joint training
- Easy to extend FedAvg or Scaffold to this setting



[Y. Chen, et. al., "Sharper Convergence Guarantees for federated learning with Partial Model Personalization", arXiv 2309.17409]

PEFT in federated learning

- Parameter-efficient fine-tuning in LLM under the Federated learning set up



- How does data heterogeneity influence FedLoRA? An open question

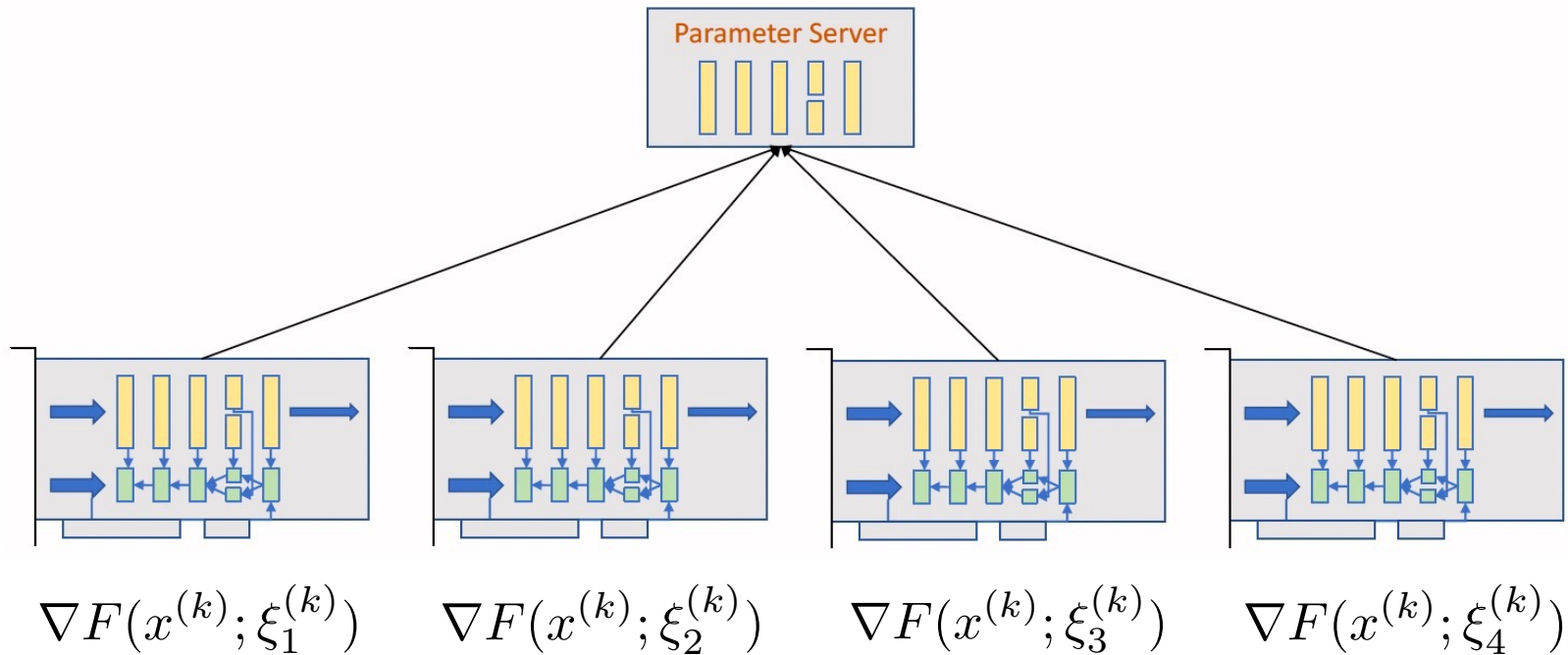
PART 05



Compressed Communication

Parallel SGD (PSGD)

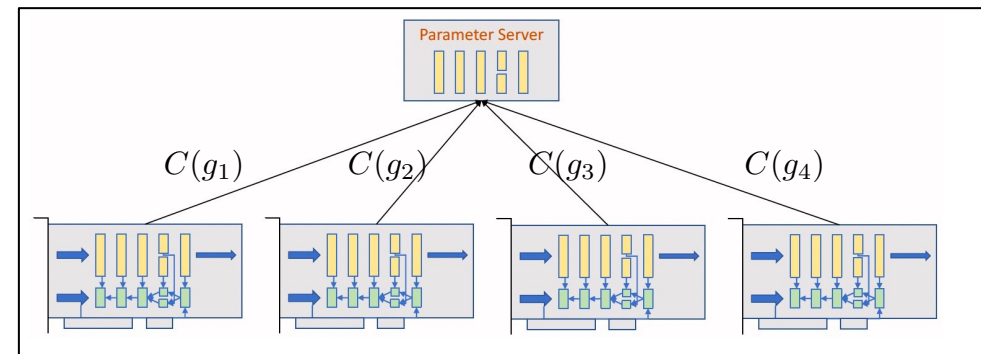
$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^n \nabla F(x^{(k)}; \xi_i^{(k)})$$



Communication compression

- A basic (but not state-of-the-art) algorithm is QSGD [Alistarh et. al., 2017]

$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$
$$x_i^{(k+1)} = x_i^{(k)} - \frac{\gamma}{n} \sum_{j=1}^n C(g_j^{(k)})$$



- $C(\cdot)$ is a compressor. It can quantize or sparsify the full gradient

Quantization

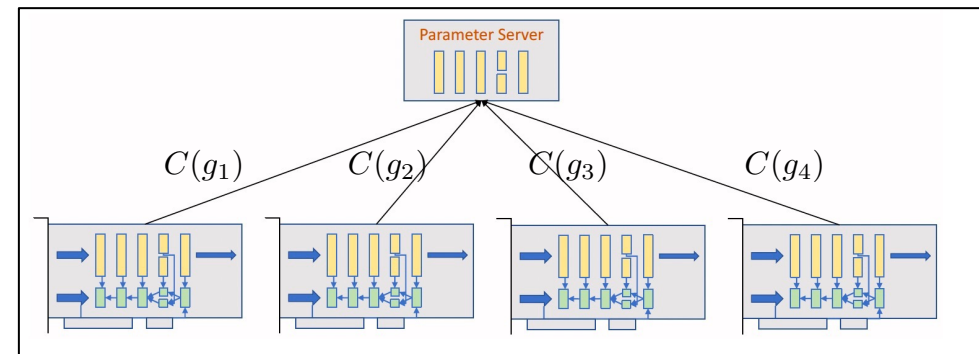


1 bit

Communication compression

- A basic (but not state-of-the-art) algorithm is QSGD [Alistarh et. al., 2017]

$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$
$$x_i^{(k+1)} = x_i^{(k)} - \frac{\gamma}{n} \sum_{j=1}^n C(g_j^{(k)})$$



- $C(\cdot)$ is a compressor. It can quantize or sparsify the full gradient

Sparsification



- Compressor class. Most compressors in literature are either **unbiased** or **contractive**
- We let \mathcal{U}_ω denote the set of unbiased compressors satisfying Assumption 3

Assumption (Unbiased compressor) The compression operator $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies

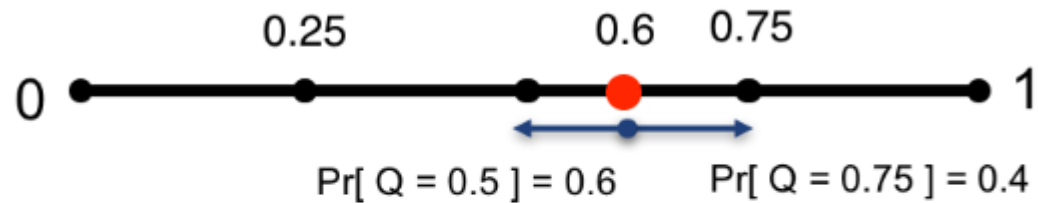
$$\mathbb{E}[C(x)] = x, \quad \mathbb{E}[\|C(x) - x\|^2] \leq \omega \|x\|^2, \quad \forall x \in \mathbb{R}^d$$

for constant $\omega \geq 0$, where the expectation is taken over the randomness of the compression operator C .

- Identity operator I (i.e. no compression) is an unbiased compressor with $\omega = 0$.

Example: random quantization

- Random quantization with 5 levels:



$$\mathbb{E}[Q] = 0.6 \times 0.5 + 0.4 \times 0.75 = 0.6$$

Unbiased:
$$\mathbb{E}[Q(x)] = \frac{Q_+(x) - x}{Q_+(x) - Q_-(x)} \cdot Q_-(x) + \frac{x - Q_-(x)}{Q_+(x) - Q_-(x)} \cdot Q_+(x) = x$$

Distributed compressed SGD

- Compressed algorithm is easy to develop with unbiased compressors

$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$
$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{j=1}^n C(g_j^{(k)})$$

- For simplicity, we assume $g_i^{(k)} = \nabla f_i(x^{(k)})$

$$\begin{aligned} \mathbb{E}[f(x^{(k+1)})] &\leq \mathbb{E}[f(x^{(k)})] + \mathbb{E}[\langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle] + \frac{L}{2} \mathbb{E} \|x^{(k+1)} - x^{(k)}\|^2 && \text{(L-smooth)} \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}[\langle \nabla f(x^{(k)}), \frac{1}{n} \sum_{i=1}^n C(\nabla f_i(x^{(k)})) \rangle] + \frac{\gamma^2 L}{2} \mathbb{E} \left\| \frac{1}{n} \sum_{i=1}^n C(\nabla f_i(x^{(k)})) \right\|^2 \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2} \sum_{i=1}^n \mathbb{E} \|C(\nabla f_i(x^{(k)}))\|^2 && \text{(Unbiased compressor)} \end{aligned}$$

Distributed compressed SGD: convergence sketch

$$\begin{aligned}\mathbb{E}[f(x^{(k+1)})] &\leq \mathbb{E}[f(x^{(k)})] + \mathbb{E}[\langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle] + \frac{L}{2} \mathbb{E} \|x^{(k+1)} - x^{(k)}\|^2 && \text{(L-smooth)} \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}[\langle \nabla f(x^{(k)}), \frac{1}{n} \sum_{i=1}^n C(\nabla f_i(x^{(k)})) \rangle] + \frac{\gamma^2 L}{2} \mathbb{E} \left\| \frac{1}{n} \sum_{i=1}^n C(\nabla f_i(x^{(k)})) \right\|^2 \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2} \sum_{i=1}^n \mathbb{E} \|C(\nabla f_i(x^{(k)}))\|^2 && \text{(Unbiased compressor)} \\ &= \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2} \sum_{i=1}^n \mathbb{E} \|C(\nabla f_i(x^{(k)})) - \nabla f_i(x^{(k)}) + \nabla f_i(x^{(k)})\|^2 \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{n^2} \sum_{i=1}^n \mathbb{E} \|C(\nabla f_i(x^{(k)})) - \nabla f_i(x^{(k)})\|^2 + \frac{\gamma^2 L}{n^2} \sum_{i=1}^n \mathbb{E} \|\nabla f_i(x^{(k)})\|^2 \\ &\leq \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{(1+\omega)\gamma^2 L}{n^2} \sum_{i=1}^n \mathbb{E} \|\nabla f_i(x^{(k)})\|^2 && \text{(Bounded compression)} \\ &= \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{(1+\omega)\gamma^2 L}{n^2} \sum_{i=1}^n \mathbb{E} \|\nabla f_i(x^{(k)}) - \nabla f(x^{(k)}) + \nabla f(x^{(k)})\|^2 \\ &\leq \mathbb{E}[f(x^{(k)})] - \left(\gamma - \frac{2(1+\omega)\gamma^2 L}{n^2} \right) \mathbb{E} \|\nabla f(x^{(k)})\|^2 + \frac{2(1+\omega)\gamma^2 L b^2}{n^2} && \text{(Bounded data hetero.)}\end{aligned}$$

Assumption [Compressed SGD assumption]

(1) Each $f_i(x)$ is L -smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y ;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

(3) Gradient dissimilarity can be upper bounded, i.e.,

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq b^2$$

Same as DSGD and Local SGD assumptions

Theorem [Compressed SGD convergence]

Under the above assumptions and with proper γ , compressed SGD with unbiased compressors will converge as follows

$$\frac{1}{K} \sum_{k=1}^K \mathbb{E} \|\nabla f(x^{(k)})\|^2 = \mathcal{O} \left(\sqrt{\frac{(1 + \omega)\sigma^2 + \omega b^2}{nK}} + \frac{1 + \omega/n}{K} \right)$$

Convergence performance can be further improved by DIANA [1] and MARINA [2]

[1] K. Mishchenko et. al., "Distributed Learning with Compressed Gradient Differences", 2019

[2] E. Gorbunov, et. al., "MARINA: Faster Non-Convex Distributed Learning with Compression", 2021

- We let \mathcal{C}_δ denote the set of unbiased compressors satisfying Assumption 4

Assumption (Contractive compressor) The compression operator $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies

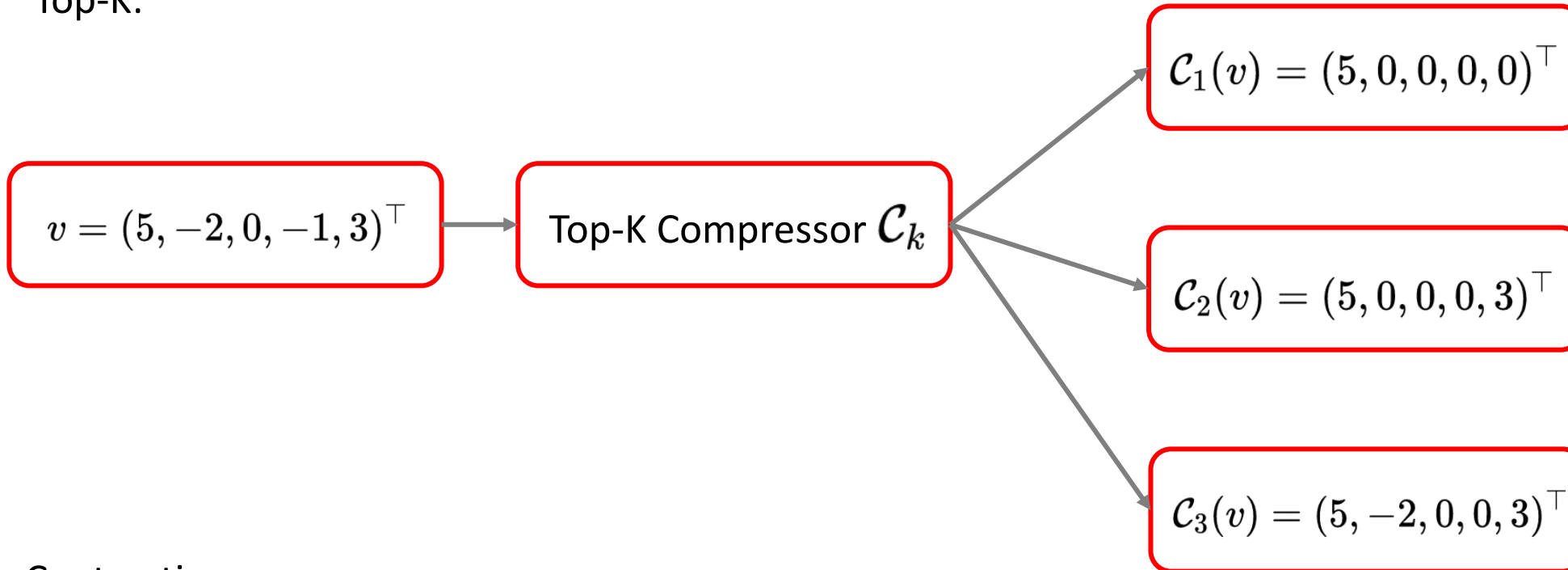
$$\mathbb{E}[\|C(x) - x\|^2] \leq (1 - \delta)\|x\|^2, \quad \forall x \in \mathbb{R}^d$$

for constant $\delta \in (0, 1]$, where the expectation is taken over the randomness of the compression operator C .

- Identity operator I (i.e. no compression) is a contractive compressor with $\delta = 1$.

Example: Top-K compressor

- Top-K:



Contractive:

$$\|\mathcal{C}_k(v) - v\|^2 = \|v\|^2 - \|\mathcal{C}_k(v)\|^2 \leq (1 - k/d)\|v\|^2, \text{ i.e., } \delta = k/d \in (0, 1]$$

Distributed compressed SGD may diverge with contractive compressor

- DC-SGD may diverge with contractive compressors

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^n C(\nabla f_i(x^{(k)}))$$

- Check the divergent examples in [1]
- The root reason is that contractive compressor introduces non-vanishing distortions

$$\mathbb{E} \|C(\nabla f_i(x^{(k)})) - \nabla f_i(x^{(k)})\|^2 \leq (1 - \delta) \|\nabla f_i(x^{(k)})\|^2$$

- Since $\nabla f_i(x^{(k)})$ will not converge to 0, $C(\nabla f_i(x^{(k)}))$ will not converge to $\nabla f_i(x^{(k)})$

[1] A. Beznosikov et.al., “On Biased Compression for Distributed Learning”, JMLR 2023

Error feedback: corrects the distortion

- A smart idea: we do not compress the gradient. Instead, we compress the **gradient difference**

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^n g_i^{(k)}$$

$$g_i^{(k+1)} = g_i^{(k)} + C(\nabla f_i(x^{(k+1)}) - g_i^{(k)})$$

- The compression distortion gets diminished, and $g_i^{(k)} \rightarrow \nabla f_i(x^{(k)})$

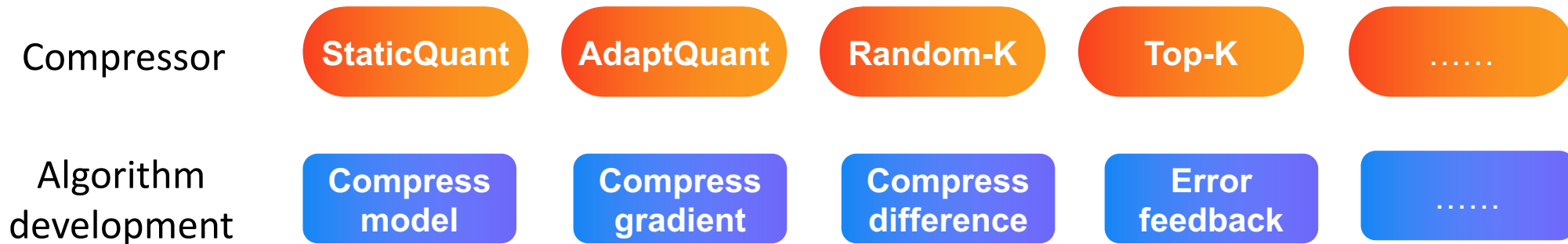
$$\begin{aligned} \mathbb{E}\|g_i^{(k+1)} - \nabla f_i(x^{(k+1)})\|^2 &= \mathbb{E}\|C(\nabla f_i(x^{(k+1)}) - g_i^{(k)}) - (\nabla f_i(x^{(k+1)}) - g_i^{(k)})\|^2 \\ &\leq (1 - \delta)\mathbb{E}\|g_i^{(k)} - \nabla f_i(x^{(k+1)})\|^2 \end{aligned}$$

- If $\nabla f_i(x^{(k)}) \rightarrow \nabla f_i(x^*)$, we will have $g_i^{(k)} \rightarrow \nabla f_i(x^{(k)})$ and hence distortion gets diminished
- This algorithm is named as Error Feedback [1]

[1] P. Richtárik, et. al., “EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback”, NeurIPS 2021

Communication compression algorithms

- There are extensive studies in distributed learning with communication compression



- The combination of different compressors, algorithms, and strategies gives rise to

Q-SGD [Alistarh et. al., 2017], Mem-SGD [Stich et. al., 2018], EF21-SGD [Fatkhullin et. al., 2021], CSER [Xie et.al., 2020], Double Squeeze [Tang et. al., 2019], Artemis [Philippenko et.al. 2022], etc.

- What is the **optimal complexity** in distributed optimization with communication compression?
- Can we develop **effective algorithms** that can attain the optimal complexity?

PART 05-1

Compressed communication: Optimal Complexity Formulation

Function class and gradient oracle class

- Function class. We let $\mathcal{F}_{\mu,L}$ denote the set of all functions satisfying Assumption 1

Assumption 1 (Smoothness and Strong Convexity) Each local objective f_i has L -Lipschitz gradient and μ -strongly convex. Moreover, we assume that $f(x^{(0)}) - \inf_{x \in \mathbb{R}^d} f(x) \leq \Delta$ with $f = \frac{1}{n} \sum_{i=1}^n f_i$.

- Gradient oracle class. Each worker accesses local gradient $\nabla f_i(x)$ via a stochastic oracle

Assumption 2 (Stochastic gradient) The gradient oracles $\{O_i : 1 \leq i \leq n\}$ satisfy

$$\mathbb{E}_{\zeta_i}[O_i(x; \zeta_i)] = \nabla f_i(x) \quad \text{and} \quad \mathbb{E}_{\zeta_i}[\|O_i(x; \zeta_i) - \nabla f_i(x)\|^2] \leq \sigma^2, \quad \forall x \in \mathbb{R}^d.$$

- Compressor class. Most compressors in literature are either **unbiased** or **contractive**
- We let \mathcal{U}_ω denote the set of unbiased compressors satisfying Assumption 3

Assump. 3 (Unbiased compressor) The compression operator $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies

$$\mathbb{E}[C(x)] = x, \quad \mathbb{E}[\|C(x) - x\|^2] \leq \omega \|x\|^2, \quad \forall x \in \mathbb{R}^d$$

for constant $\omega \geq 0$, where the expectation is taken over the randomness of the compression operator C .

- Identity operator I (i.e. no compression) is an unbiased compressor with $\omega = 0$.

- Compressor class. Most compressors in literature are either **unbiased** or **contractive**
- We let \mathcal{C}_δ denote the set of unbiased compressors satisfying Assumption 4

Assump. 4 (Contractive compressor) The compression operator $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies

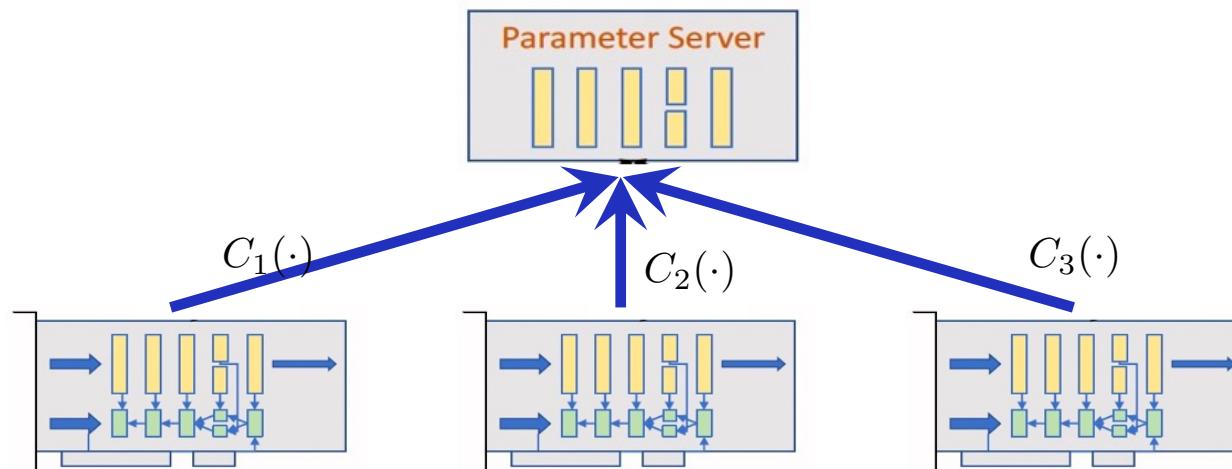
$$\mathbb{E}[\|C(x) - x\|^2] \leq (1 - \delta)\|x\|^2, \quad \forall x \in \mathbb{R}^d$$

for constant $\delta \in (0, 1]$, where the expectation is taken over the randomness of the compression operator C .

- Identity operator I (i.e. no compression) is a contractive compressor with $\delta = 1$.

Algorithm Class

- Workers communicate directly with a central server. All iterations are synchronized.
- Each worker $i \in \{1, \dots, n\}$ is endowed with C_i .
- First-order algorithms



Complexity metric

- Let \hat{x}_A^t denote the output of algorithm A after t communication rounds
- We define the complexity metric as

$$T_\epsilon(A, \{(f_i, C_i)\}_{i=1}^n) = \min \left\{ t \in \mathbb{N} : \mathbb{E}[f(\hat{x}_A^t)] - \min_x f(x) \leq \epsilon \right\}$$

- It indicates the **smallest** number of iterations to achieve an ϵ - accurate solution

Find optimal complexity

- The problem to find optimal complexity can be formulated into the problem

$$\inf_{A \in \mathcal{A}} \sup_{\{C_i\}_{i=1}^n \subseteq \mathcal{C}} \sup_{\{O_i\}_{i=1}^n \subseteq \mathcal{O}_{\sigma^2}} \sup_{\{f_i\}_{i=1}^n \subseteq \mathcal{F}_{\mu, L}} T_\epsilon(A, \{(f_i, C_i)\}_{i=1}^n)$$

- In other words, given a class of functions $\mathcal{F}_{\mu, L}$, gradient oracles \mathcal{O}_{σ^2} , compressors \mathcal{C} (being \mathcal{C}_δ or \mathcal{U}_ω), the formulation seeks the optimal algorithm and the convergence complexity it has.

Find optimal complexity

- The problem to find optimal complexity can be formulated into the problem

$$\inf_{A \in \mathcal{A}} \sup_{\{C_i\}_{i=1}^n \subseteq \mathcal{C}} \sup_{\{O_i\}_{i=1}^n \subseteq \mathcal{O}_{\sigma^2}} \sup_{\{f_i\}_{i=1}^n \subseteq \mathcal{F}_{\mu, L}} T_\epsilon(A, \{(f_i, C_i)\}_{i=1}^n)$$

- Very challenging to directly solve the above problem
- We will establish the lower bound first, then the upper bound, and finally show they match each other
- The algorithm to achieve the lower bound is optimal

PART 05-2

Compressed Communication: Find Lower Bound

Theorem 1 (**Unbiased** compression)

For every $\Delta, L > 0, n \geq 2, \omega \geq 0, \sigma > 0, T \geq (1 + \omega)^2$, there exists a set of local loss functions $\{f_i\}_{i=1}^n \subseteq \mathcal{F}_{\mu, L}$, stochastic gradient oracles $\{O_i\}_{i=1}^n \subseteq \mathcal{O}_{\sigma^2}$, ω -unbiased compressors $\{C_i\}_{i=1}^n \subseteq \mathcal{U}_{\omega}$, such that for any algorithm $A \in \mathcal{A}$ starting from a given constant $x^{(0)}$, the lower bound complexity is given by

$$\Omega \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right),$$

where ϵ is the desired accuracy.

X. Huang et. al., "Lower Bounds and Nearly Optimal Algorithms in Distributed Learning with Communication Compression", NeurIPS 2022

Y. He, et. al., "Lower Bounds and Accelerated Algorithms in Distributed Stochastic Optimization with Communication Compression", 2023

Consistency with existing lower bound

$$\Omega \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$$

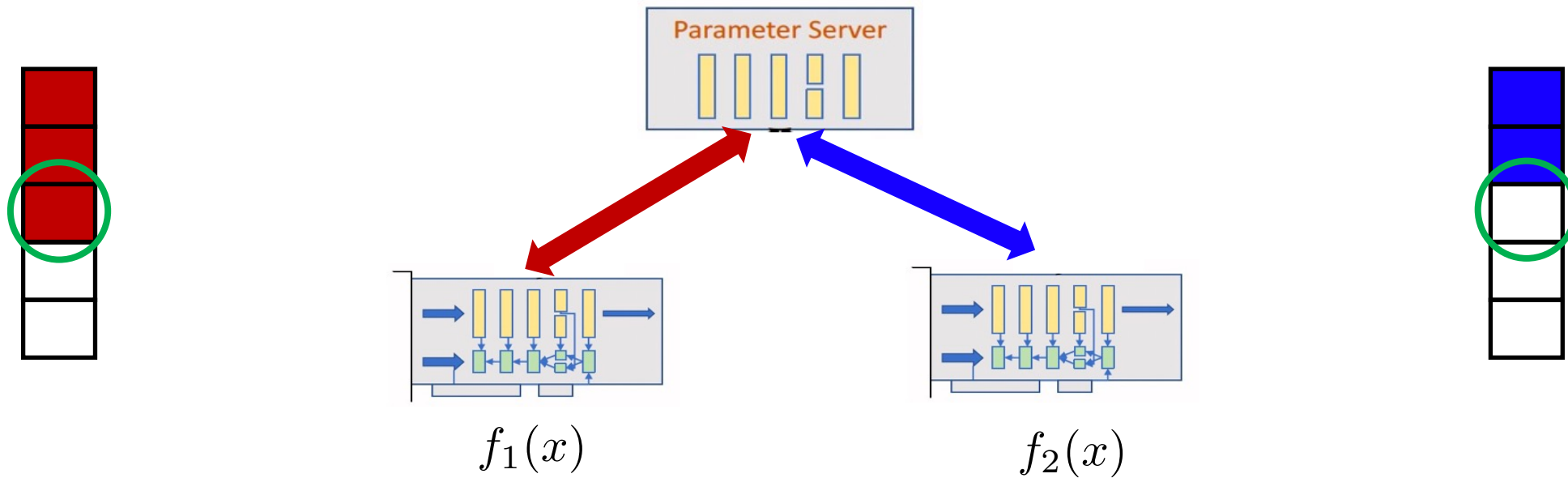
σ^2 is the gradient noise, n is the number of nodes, and ω gauges the compression ratio

- When $n = 1$ and $\omega = 0$, it recovers the lower bound in stochastic strongly-convex optimization
- When $n = 1$, $\omega = 0$ and $\sigma^2 = 0$, it recovers Nesterov lower bound

$$(n = 1, \omega = 0) \quad \Omega \left(\frac{\sigma^2}{\mu \epsilon} + \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right) \quad (n = 1, \omega = 0, \sigma = 0) \quad \Omega \left(\sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$$

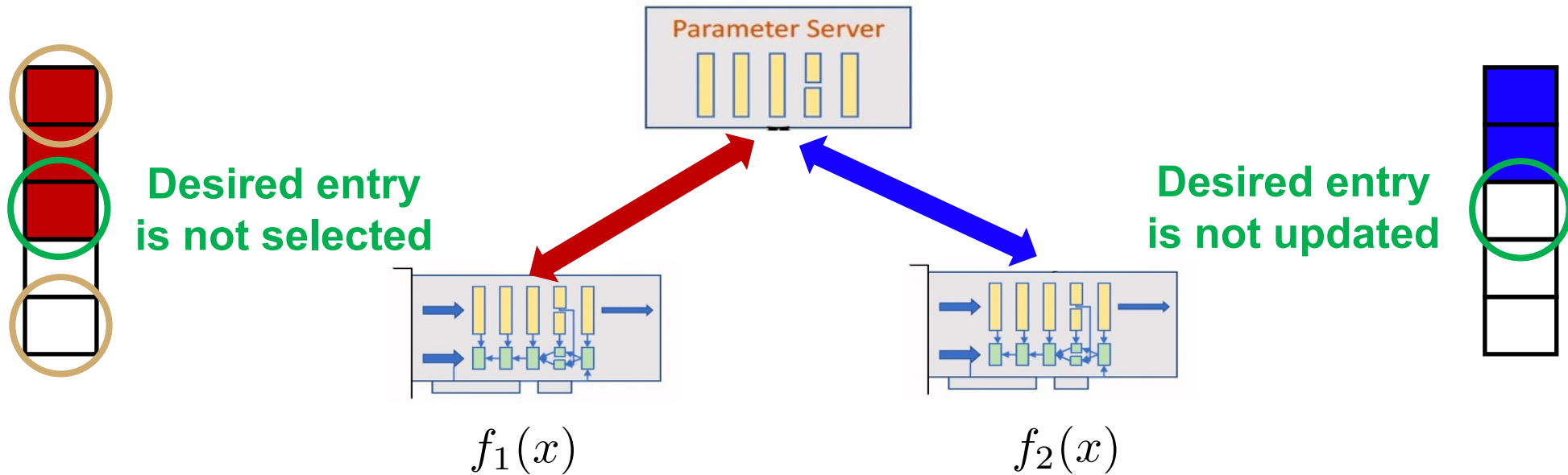
Distributed optimization with compression

- We construct adversarial objective functions
- The odd element in $f_2(x)$ can only be updated by receiving corresponding element from worker 1



Intuition behind lower bound

- With compression, the desired entry may not be able to communicate

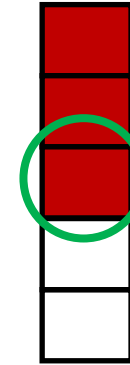


- It explains why communication compression converges slower than vanilla distributed SGD

Intuition behind lower bound

- What's the probability that the desired entry is sent?

- We constructed an adversarial compressor: **rand-s**



Desired entry
is not selected

- In rand-s compressor, s elements will be uniformly randomly chosen from all d elements

- Rand-s is an unbiased compressor (after scale) with $\omega = \frac{d}{s} - 1$

- The desired entry is sent with probability $s/d = (1 + \omega)^{-1}$

Intuition behind lower bound

- In expectation, the desired entry will experience $(1 + \omega)$ rounds to be sent
- Recall the lower bound complexity of standard distributed optimization as

$$\Omega \left(\sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$$

- Since sending the desired entry require $(1 + \omega)$ rounds, the lower bound with compression is

$$\Omega \left((1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$$

Theorem 2 (Contractive compression)

For every $\Delta, L > 0, n \geq 2, \omega \geq 0, \sigma > 0, T \geq (1 + \omega)^2$, there exists a set of local loss functions $\{f_i\}_{i=1}^n \subseteq \mathcal{F}_{\mu, L}$, stochastic gradient oracles $\{O_i\}_{i=1}^n \subseteq \mathcal{O}_{\sigma^2}$, ω -unbiased compressors $\{C_i\}_{i=1}^n \subseteq \mathcal{C}_\delta$, such that for any algorithm $A \in \mathcal{A}$ starting from a given constant $x^{(0)}$, it holds that

$$\mathbb{E}[f(\hat{x}) - f^*] = \Omega \left(\frac{\sigma^2}{\mu n \epsilon} + \frac{1}{\delta} \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right),$$

PART 05-3

Compressed communication: Algorithm to Attain Lower Bounds

Optimal algorithm for distributed optimization

- Recall the problem $\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$, where $f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F(x; \xi_i)$.
- We first consider the scenario in which the true gradient $\nabla f_i(x)$ is known and no compression is used
- The optimal algorithm is Nesterov Acceleration. For $k=1,2,\dots,T$

$$\left. \begin{aligned} z^k &= \frac{1}{\gamma} x^k + \left(\frac{1}{p} - \frac{1}{\gamma} \right) x^{k-1} + \left(1 - \frac{1}{p} \right) z^{k-1} \\ y^k &= \left(1 - \frac{\gamma}{p} \right) x^k + \frac{\gamma}{p} z^k \end{aligned} \right\} \text{(extrapolation)}$$

$$g_i^k = \nabla f_i(y^k) \quad \text{(true gradient)}$$

$$g^k = \frac{1}{n} \sum_{i=1}^n g_i^k \quad \text{(precise communication)}$$

$$x^{k+1} = y^k - \frac{\eta}{p} g^k \quad \text{(gradient descent)}$$

Extension to stochastic and compressed scenario

- A trivial extension: For $k = 1, 2, \dots, T$

$$\left. \begin{aligned} z^k &= \frac{1}{\gamma} x^k + \left(\frac{1}{p} - \frac{1}{\gamma} \right) x^{k-1} + \left(1 - \frac{1}{p} \right) z^{k-1} \\ y^k &= \left(1 - \frac{\gamma}{p} \right) x^k + \frac{\gamma}{p} z^k \end{aligned} \right\} \text{(extrapolation)}$$

$$g_i^k = \nabla F(y^k; \xi_i^k) \quad \text{(stochastic gradient)}$$

$$g^k = \frac{1}{n} \sum_{i=1}^n C_i(g_i^k) \quad \text{(communication compression)}$$

$$x^{k+1} = y^k - \frac{\eta}{p} g^k \quad \text{(gradient descent)}$$

- Does not work well. Stochastic gradient and compression introduces too much noise.

Upper bound and nearly-optimal algorithms

- A refined algorithm: For $k = 1, 2, \dots, T/R$

$$\left. \begin{aligned} z^k &= \frac{1}{\gamma} x^k + \left(\frac{1}{p} - \frac{1}{\gamma} \right) x^{k-1} + \left(1 - \frac{1}{p} \right) z^{k-1} \\ y^k &= \left(1 - \frac{\gamma}{p} \right) x^k + \frac{\gamma}{p} z^k \end{aligned} \right\} \text{(extrapolation)}$$

$$g_i^k = \frac{1}{R} \sum_{r=1}^R \nabla F(y^k; \xi_i^{(k,r)}) \quad \text{(large-batch stochastic gradient)}$$

$$g^k = \frac{1}{n} \sum_{i=1}^n \text{MSC}(g_i^k, R) \quad \text{(multi-step compression)}$$

$$x^{k+1} = y^k - \frac{\eta}{p} g^k \quad \text{(gradient descent)}$$

- The communication and sample budget are the same as previous algorithms

Upper bound and nearly-optimal algorithms

- Multi-step compression (MSC) protocol:

Input: Original vector v and compression round R

Initialize $v^{(0)} = 0$;

for $r = 0, \dots, R - 1$ **do**

 compress $v - v^{(r)}$ into $c^{(r)} = C(v - v^{(r)})$

 send $c^{(r)}$ to the target

 update $v^{(r+1)} = v^{(r)} + c^{(r)}$

end for

- With MSC, we can prove

$$v^{(R)} = \sum_{r=0}^{R-1} c^{(r)}$$

$$\mathbb{E} \|v^{(R)} - v\|^2 \leq (1 - \delta)^R \|v\|^2$$

- It holds that

$$v^{(R)} \rightarrow v \text{ as } R \rightarrow \infty$$

Upper bound and nearly-optimal algorithms

- A refined algorithm NEOLITHIC: For $k = 1, 2, \dots, T/R$

$$\left. \begin{aligned} z^k &= \frac{1}{\gamma} x^k + \left(\frac{1}{p} - \frac{1}{\gamma} \right) x^{k-1} + \left(1 - \frac{1}{p} \right) z^{k-1} \\ y^k &= \left(1 - \frac{\gamma}{p} \right) x^k + \frac{\gamma}{p} z^k \end{aligned} \right\} \text{(extrapolation)}$$

$$g_i^k = \frac{1}{R} \sum_{r=1}^R \nabla F(y^k; \xi_i^{(k,r)}) \quad \text{(large-batch stochastic gradient)}$$

$$g^k = \frac{1}{n} \sum_{i=1}^n \text{MSC}(g_i^k, R) \quad \text{(multi-step compression)}$$

$$x^{k+1} = y^k - \frac{\eta}{p} g^k \quad \text{(gradient descent)}$$

- As $R \rightarrow \infty$, the refined algorithm will approach to vanilla Nesterov; the introduced noise can be controlled

Theorem 3 (Upper bound)

(Informal) When utilizing unbiased compressors, the algorithm converges at

$$\mathbb{E}[f(x^K) - f^*] = \tilde{\mathcal{O}} \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right),$$

where $\tilde{\mathcal{O}}(\cdot)$ hides logarithm terms that are independent of ϵ . When using contractive compressors, the algorithm converges at

$$\mathbb{E}[f(x^K) - f^*] = \tilde{\mathcal{O}} \left(\frac{\sigma^2}{\mu n \epsilon} + \frac{1}{\delta} \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right).$$

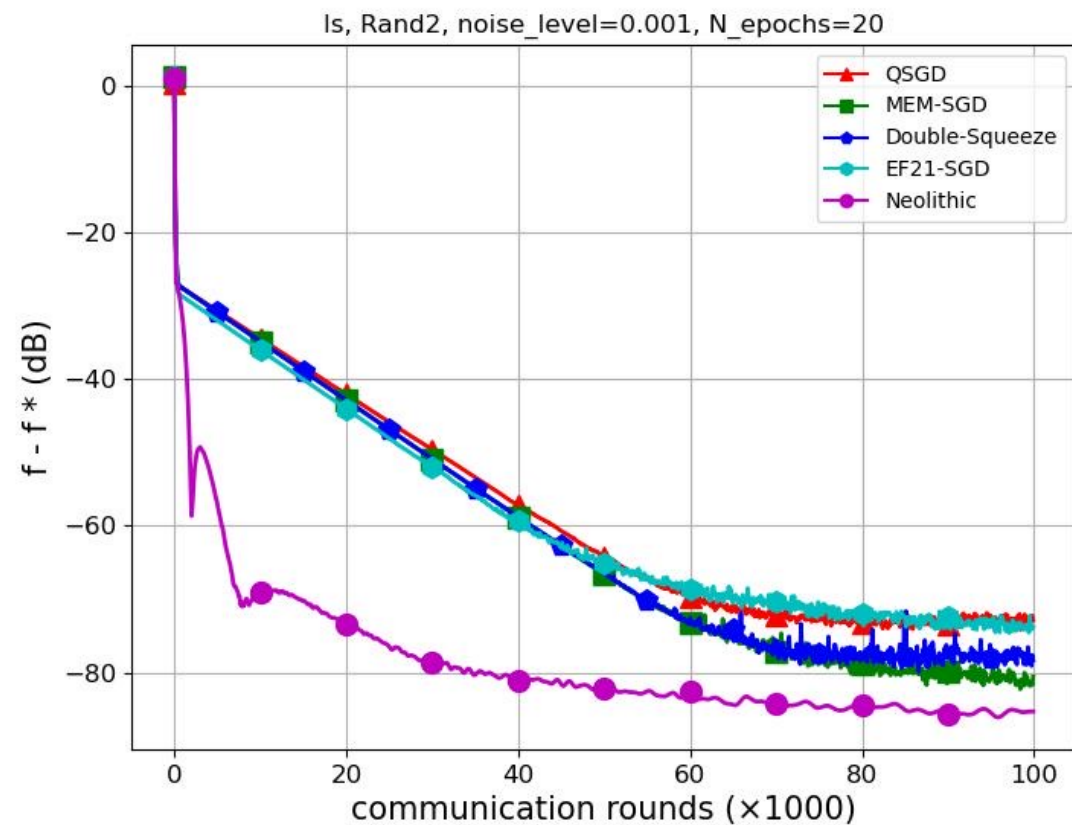
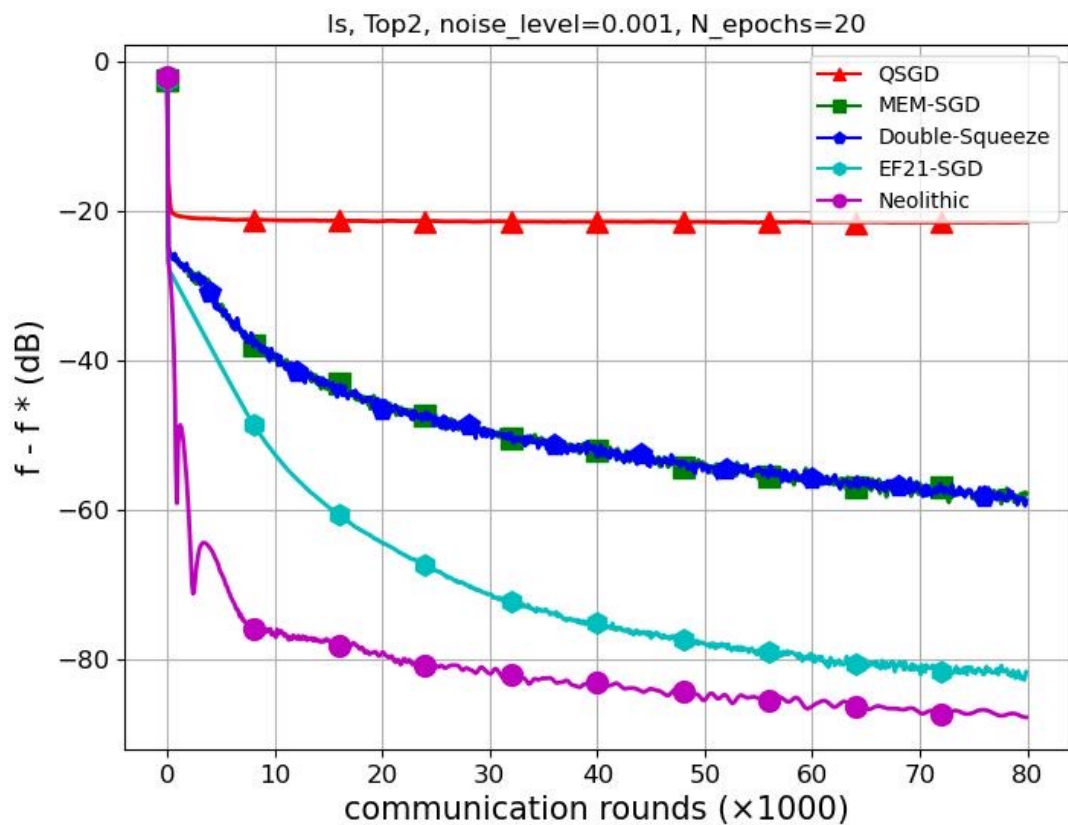
$$\inf_{A \in \mathcal{A}} \sup_{\{C_i\}_{i=1}^n \subseteq \mathcal{C}} \sup_{\{O_i\}_{i=1}^n \subseteq \mathcal{O}_{\sigma^2}} \sup_{\{f_i\}_{i=1}^n \subseteq \mathcal{F}_{\mu, L}} T_\epsilon(A, \{(f_i, C_i)\}_{i=1}^n)$$

- The lower bound and upper bound are nearly matched

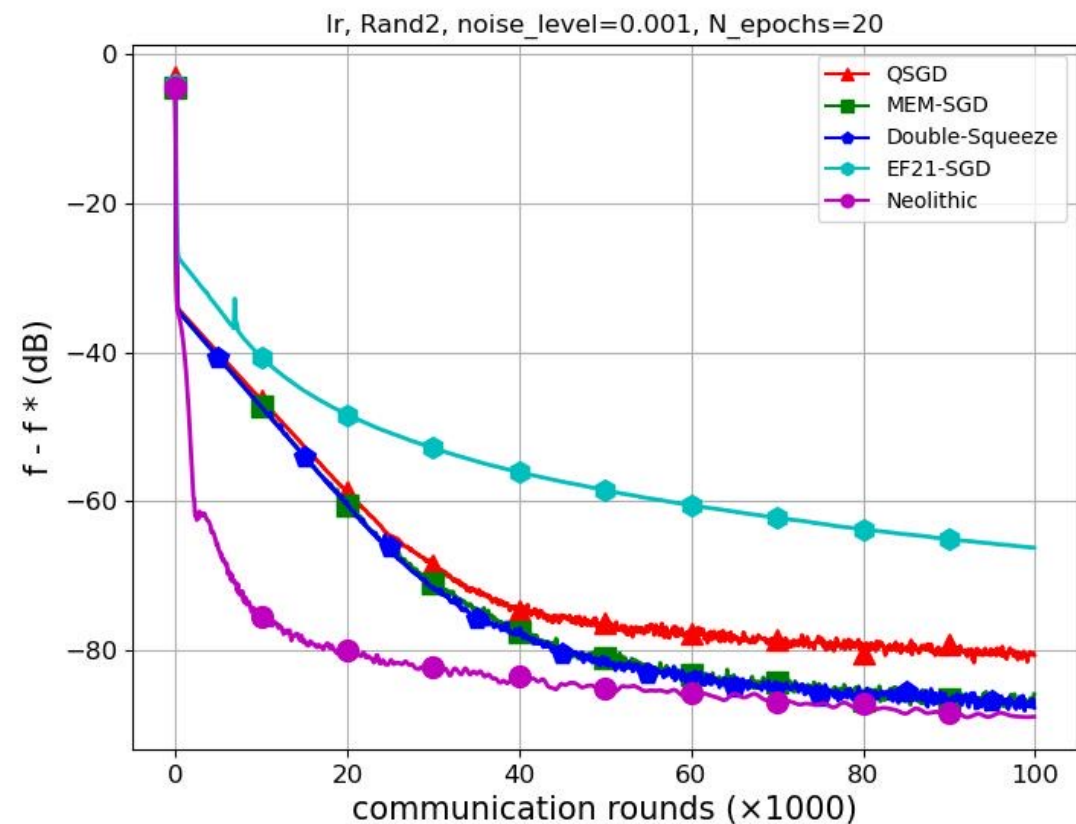
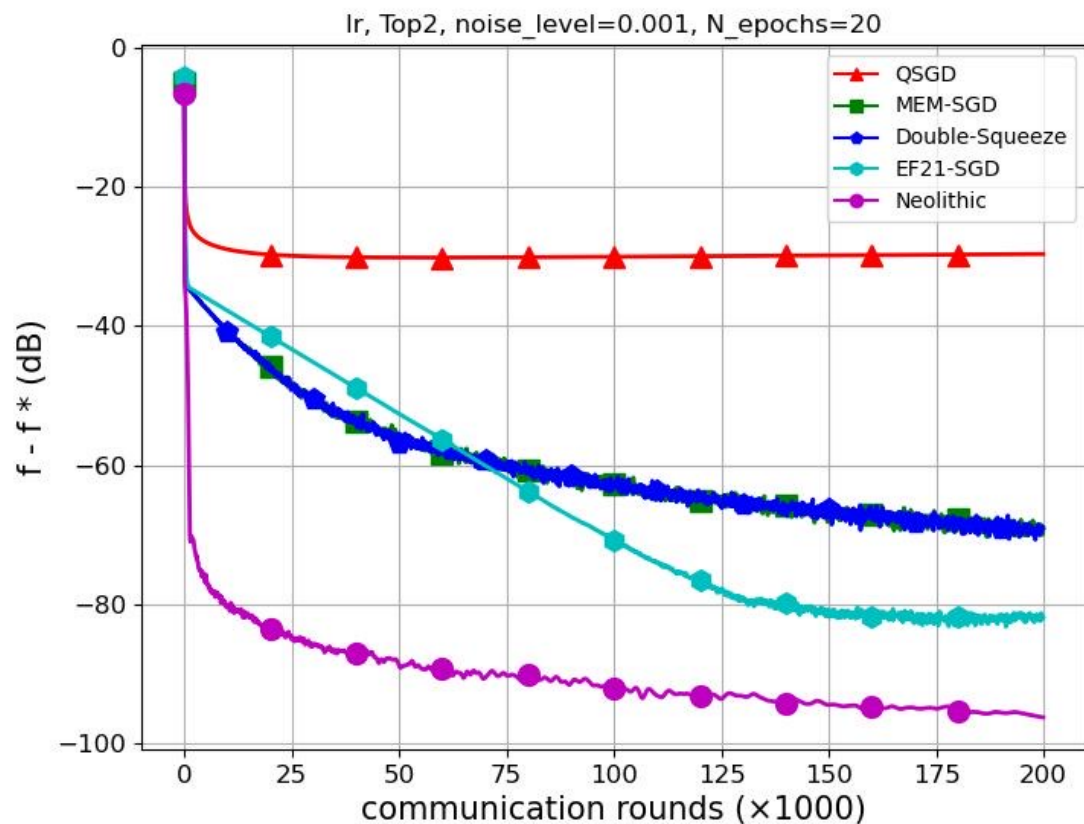
$$\begin{aligned} \mathbb{E}[f(\hat{x}) - f^*] &= \Omega \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right) \\ &= \tilde{O} \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right) \end{aligned}$$

- **The established lower bound is tight. Our algorithm is optimal.**

Least square with synthetic data



Logistic regression with synthetic data



- We also study generally-convex and non-convex scenarios

Unbiased compressor

| Method | NC | GC | SC |
|--------|--|---|---|
| L.B. | $\Omega\left(\frac{L\Delta_f\sigma^2}{n\varepsilon^2} + \frac{(1+\omega)L\Delta_f}{\varepsilon}\right)$ | $\Omega\left(\frac{\Delta_x\sigma^2}{n\varepsilon^2} + \frac{(1+\omega)\sqrt{L\Delta_x}}{\sqrt{\varepsilon}}\right)$ | $\Omega\left(\frac{\sigma^2}{\mu n\varepsilon} + (1+\omega)\sqrt{\frac{L}{\mu}} \ln\left(\frac{\mu\Delta_x}{\varepsilon}\right)\right)$ |
| Ours | $\tilde{O}\left(\frac{L\Delta_f\sigma^2}{n\varepsilon^2} + \frac{(1+\omega)L\Delta_f}{\varepsilon}\right)$ | $\tilde{O}\left(\frac{\Delta_x\sigma^2}{n\varepsilon^2} + \frac{(1+\omega)\sqrt{L\Delta_x}}{\sqrt{\varepsilon}}\right)$ | $\tilde{O}\left(\frac{\sigma^2}{\mu n\varepsilon} + (1+\omega)\sqrt{\frac{L}{\mu}} \ln\left(\frac{1}{\varepsilon}\right)\right)$ |

- Lower bound and upper bound are **nearly matched** (up to logarithm terms)
- The lower bound is tight. Our proposed NEOLITHIC is nearly optimal**

- We also study generally-convex and non-convex scenarios

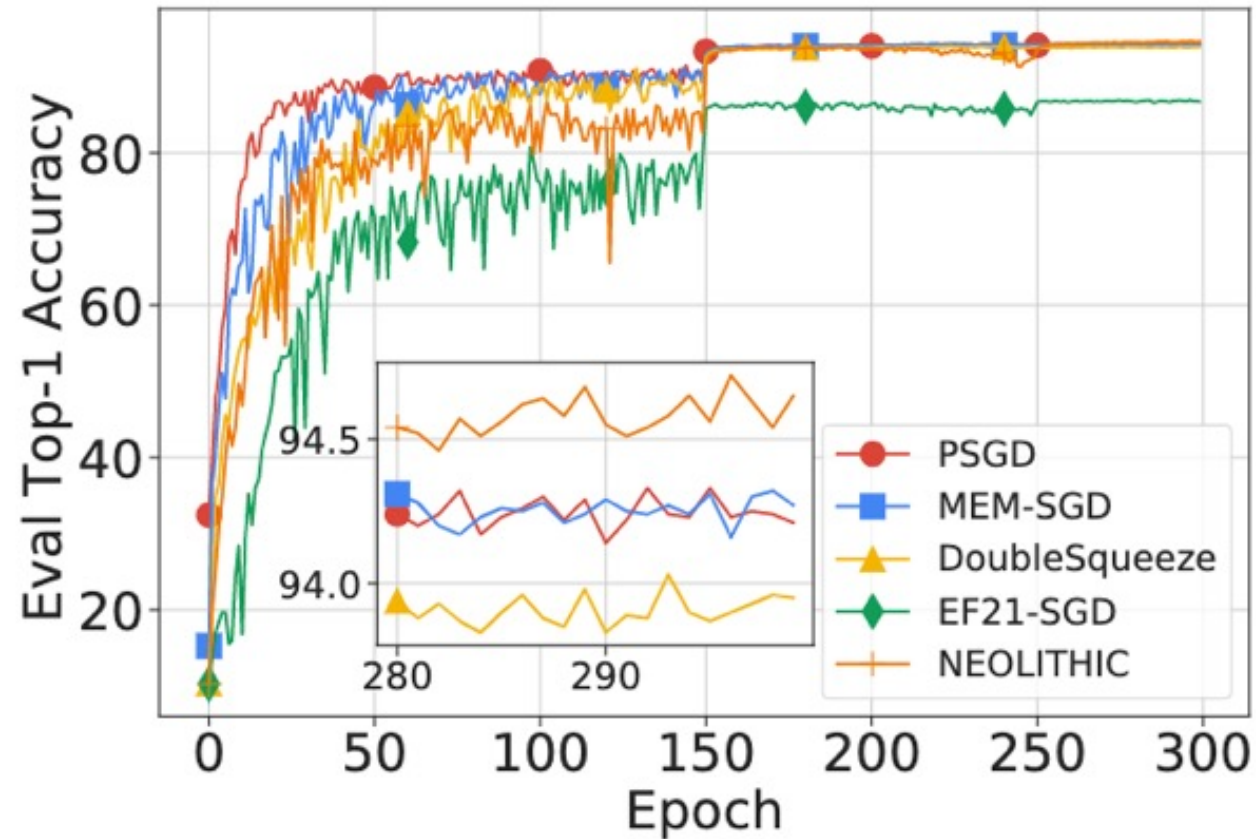
Contractive compressor

| Method | NC | GC | SC |
|--------|---|---|--|
| L.B. | $\Omega\left(\frac{L\Delta_f\sigma^2}{n\epsilon^2} + \frac{L\Delta_f}{\delta\epsilon}\right)$ | $\Omega\left(\frac{\Delta_x\sigma^2}{n\epsilon^2} + \frac{\sqrt{L\Delta_x}}{\delta\sqrt{\epsilon}}\right)$ | $\Omega\left(\frac{\sigma^2}{\mu n\epsilon} + \frac{1}{\delta}\sqrt{\frac{L}{\mu}}\ln\left(\frac{\mu\Delta_x}{\epsilon}\right)\right)$ |
| Ours | $\tilde{O}\left(\frac{L\Delta_f\sigma^2}{n\epsilon^2} + \frac{L\Delta_f}{\delta\epsilon}\ln\left(\frac{1}{\epsilon}\right)\right)^\ddagger$ | $\tilde{O}\left(\frac{\Delta_x\sigma^2}{n\epsilon^2} + \frac{\sqrt{L\Delta_x}}{\delta\sqrt{\epsilon}}\ln\left(\frac{1}{\epsilon}\right)\right)$ | $\tilde{O}\left(\frac{\sigma^2}{\mu n\epsilon} + \frac{1}{\delta}\sqrt{\frac{L}{\mu}}\ln\left(\frac{1}{\epsilon}\right)\right)$ |

- Lower bound and upper bound are **nearly matched** (up to logarithm terms)
- The lower bound is tight. Our proposed NEOLITHIC is nearly optimal**

Deep learning experiments

- 8 workers, **1% compression ratio** (top-k compressors), minibatch=128, R=2, ResNet18/ResNet20



Deep learning experiments

Cifar10; 8 workers; **5% compression ratio**

| METHODS | RESNET18 | RESNET20 |
|----------------|------------------------------------|------------------------------------|
| PSGD | 93.99 \pm 0.52 | 91.62 \pm 0.13 |
| MEM-SGD | 94.35 \pm 0.01 | 91.27 \pm 0.08 |
| DOUBLE-SQUEEZE | 94.11 \pm 0.14 | 90.73 \pm 0.02 |
| EF21-SGD | 87.37 \pm 0.49 | 65.82 \pm 4.86 |
| NEOLITHIC | 94.63 \pm 0.09 | 91.43 \pm 0.10 |

A brief summary

- Compression is critical to save communication in distributed learning
- We establish the optimal complexity for distributed learning with communication compression
- We propose NEOLITHIC to nearly attain this optimal complexity
- Our results hold for non-convex, generally-convex, and strongly-convex scenarios

The end of the story? Not Yet !

Compressed algorithms converge slower

- The optimal iteration complexity of distributed algorithms with unbiased compression

(With compression) $\Theta \left(\frac{\sigma^2}{\mu n \epsilon} + (1 + \omega) \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$

- In contrast, the accelerated distributed SGD without any compression achieves optimal complexity

(Without compression) $\Theta \left(\frac{\sigma^2}{\mu n \epsilon} + \sqrt{\frac{L}{\mu}} \ln \left(\frac{1}{\epsilon} \right) \right)$

- Algorithm with compression needs **more rounds of communication** than without compression
- But compression **saves communication per round**

Can compression save total communication?

Optimal total communication cost of compressed algorithms



- Recall the definition of unbiased compression

Assump. 3 (Unbiased compressor) The compression operator $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies

$$\mathbb{E}[C(x)] = x, \quad \mathbb{E}[\|C(x) - x\|^2] \leq \omega \|x\|^2, \quad \forall x \in \mathbb{R}^d$$

for constant $\omega \geq 0$, where the expectation is taken over the randomness of the compression operator C .

- It typically holds that $\omega \gg 1$

Optimal total communication cost of compressed algorithms

- For compressed algorithm with unbiased compression, the following proposition establishes the lower bound of the **per-iteration communication cost** [1]

Proposition 1

Let d be the dimension of the input variable x . For any compressor C satisfying Assumption 1, the per-round communication cost of $C(x)$ is lower bounded by

$$\Omega\left(\frac{d}{1+\omega}\right)$$

This lower bound is **tight** and can be achieved by random-K compressor

- The more aggressive the compressor is, the more per-round communication cost it saves

Y. He, et. al., "Unbiased Compression Saves Communication in Distributed Optimization: When and How Much?", NeurIPS 2023

Optimal total communication cost of non-compressed algorithms

- The **optimal total communication cost (TCC)** of algorithms without communication compression

$$\begin{aligned} \text{TCC} &= \Theta \left(\overset{\text{Per-round comm.}}{\color{blue}d} \cdot \left[\frac{\sigma^2}{\mu n \epsilon} + \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\epsilon} \right) \right] \right) \\ &= \Theta \left(\frac{\sigma^2 d}{\mu n \epsilon} + d \sqrt{\frac{L}{\mu}} \log \left(\frac{1}{\epsilon} \right) \right) \end{aligned}$$

number of comm. rounds

Compression v.s. non-compression

- Recall the optimal total communication cost of algorithms with compression and non-compression

Unbiased compression

$$\Theta \left(\frac{d\sigma^2}{(1+\omega)\mu n\epsilon} + d\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right) \right)$$

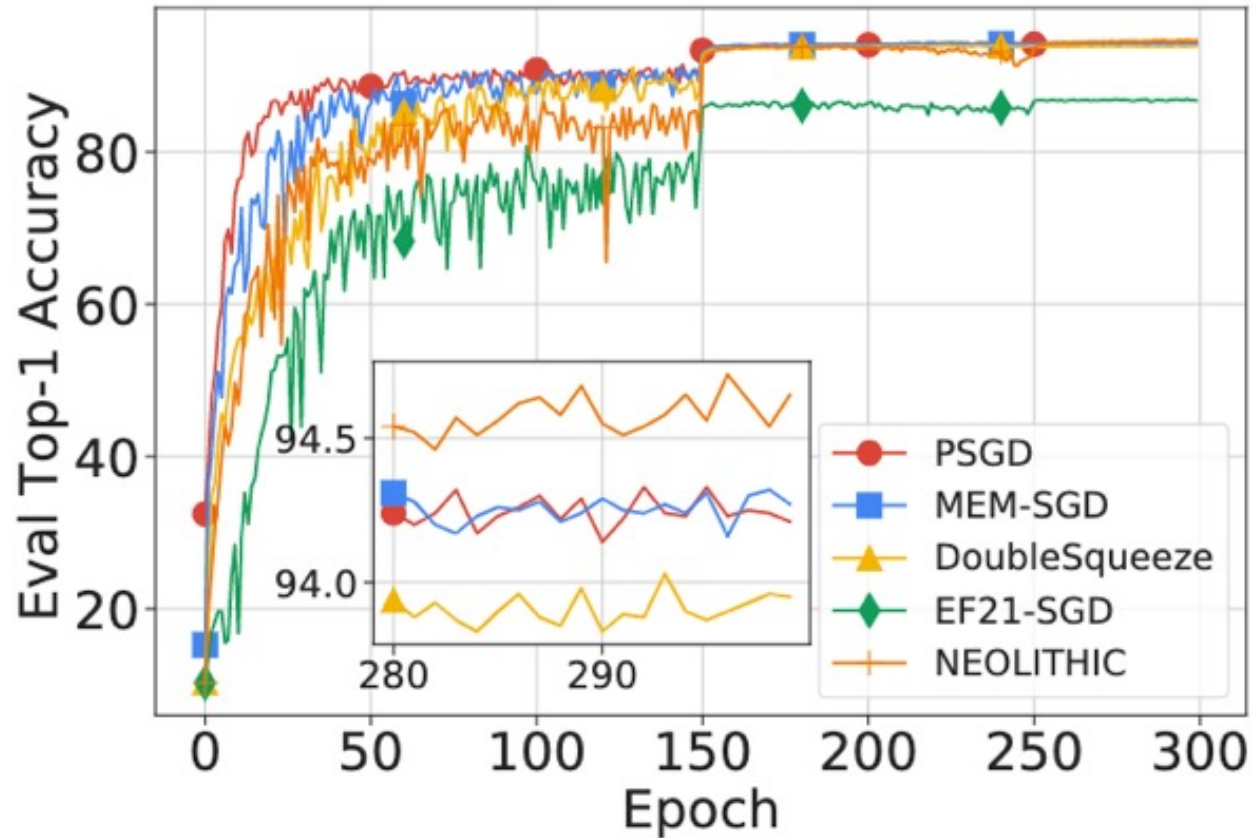
Non-compression

$$\Theta \left(\frac{\sigma^2 d}{\mu n\epsilon} + d\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right) \right)$$

- When noise σ is large or ϵ is small, the first term dominates; **compression saves communication!**
- When $\sigma \rightarrow 0$, the second term dominates; **compression does not save communication!**

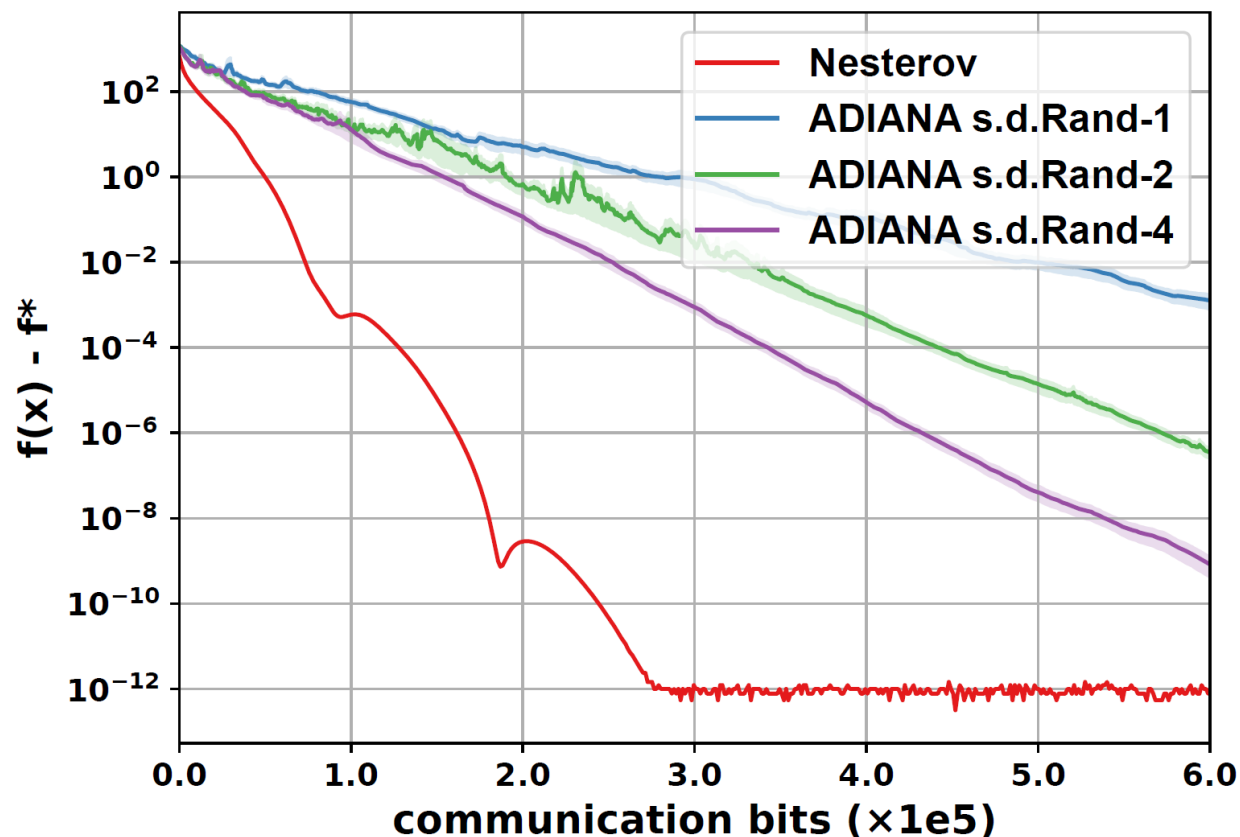
Compression saves communication in stochastic optimization

- 8 workers, **1% compression ratio** (top-k compressors), minibatch=128, R=2, ResNet18/ResNet20



Use 1% communication to reach the same accuracy

Compression does not communication in deterministic optimization



Deterministic least square

Unbiased compressor **does not save** communication

Summary

Compression saves total communication cost when

- Gradient noise is large
- A very accurate solution is required

Compression cannot save total communication cost when

- No gradient noise
- A less accurate solution is required

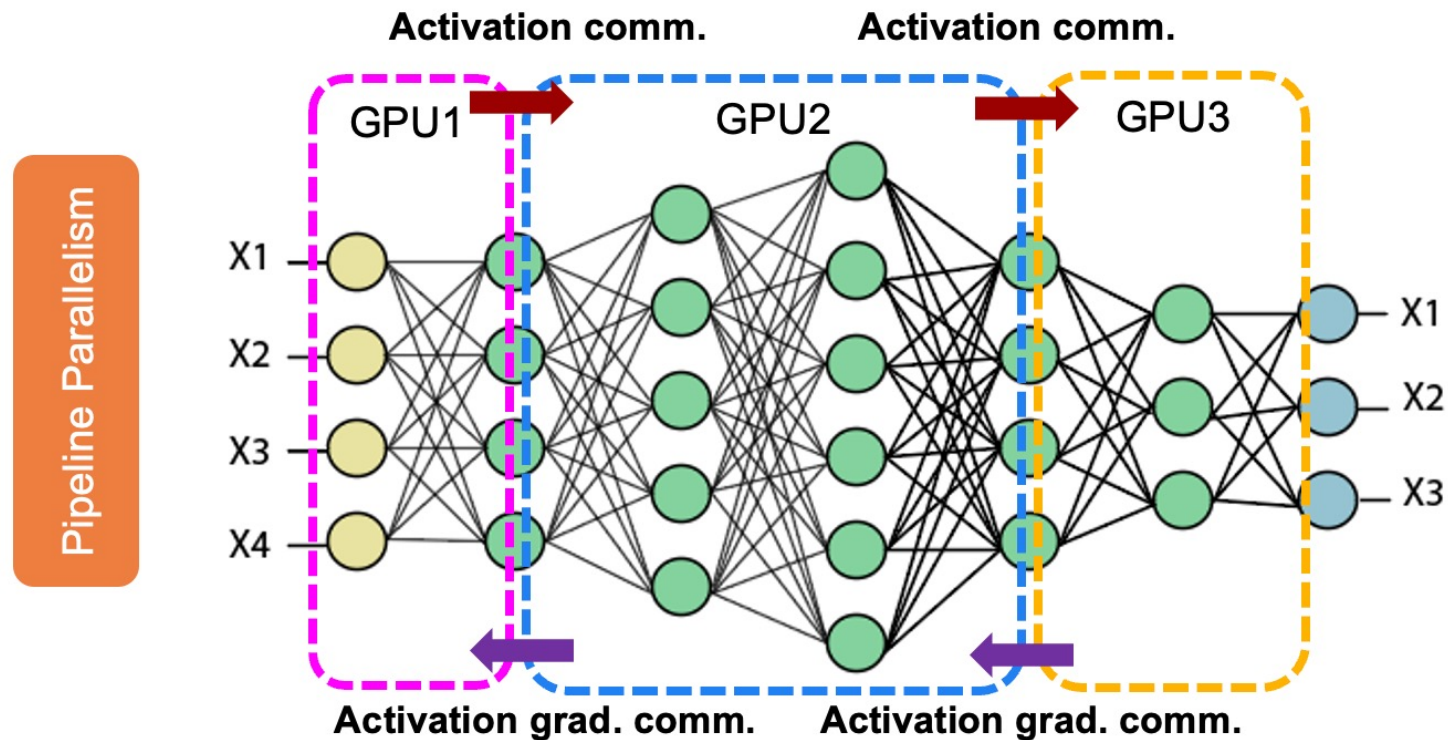
PART 06



Discussions

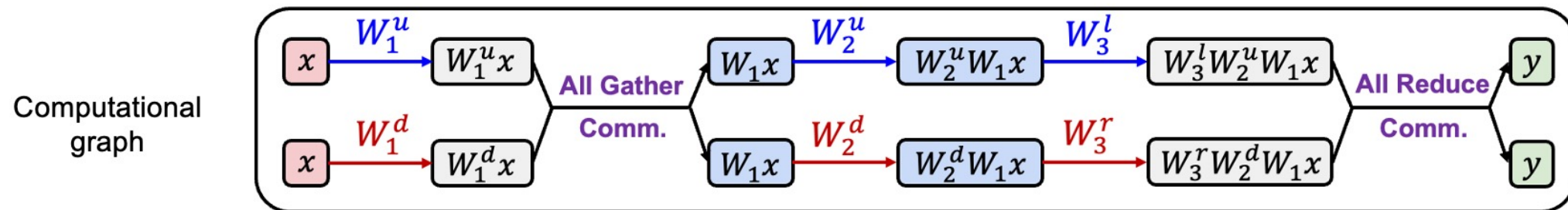
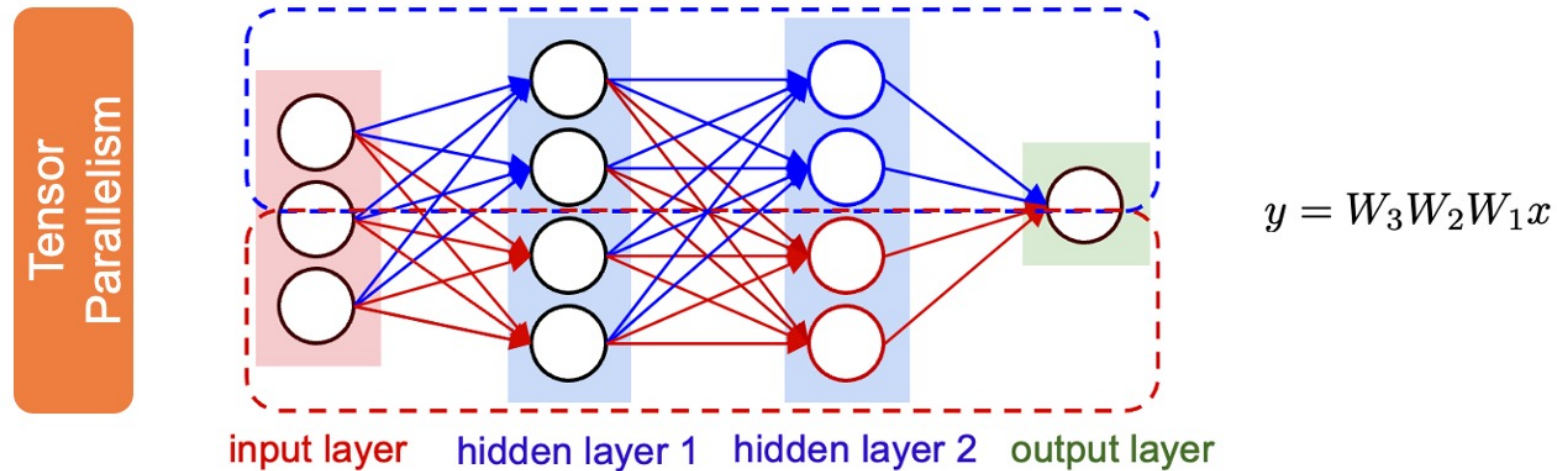
Too much data parallel! How about model parallel?

- 90% of distributed machine learning algorithms are on data parallel (as we discussed in our talk)
- Roughly speaking, communication issue for data parallelism is almost resolved
- However, communication overhead for model parallelism is almost untouched



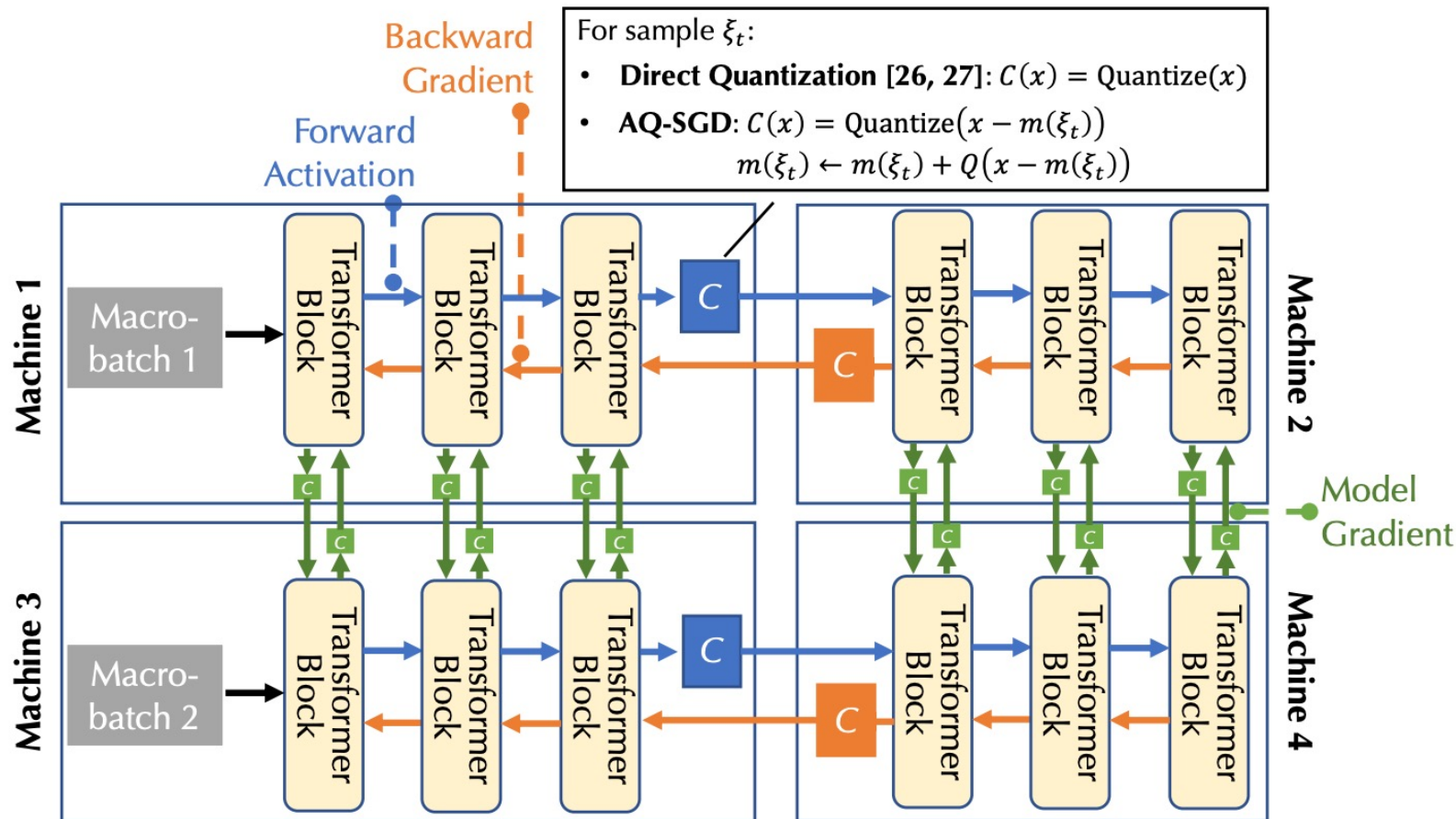
Too much data parallel! How about model parallel?

- However, communication overhead for model parallelism is almost untouched



Communication compression in pipeline parallelism

- A very recent work [1] studies activation compression in pipeline parallelism



[1] J. Wang, et. al., “Fine-tuning Language Models over Slow Networks using Activation Quantization with Guarantees”, NeurIPS 2022

Communication compression in pipeline parallelism

- Problem formulation

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} F(b(a(\xi, x^{(a)}), x^{(b)})),$$

- When using communication compression, distortion is introduced **during** forward-backward process, which is fundamentally different from data parallelism that introduces distortion **after** FB process

$$x_{t+1} = x_t - \gamma \cdot (g_\xi(x_t) + \Delta_\xi(x_t)),$$

where the distortion is defined as

$$\Delta_\xi^{(Q)}(x_t) = Q(\nabla_a(f \circ b)|_{(m(\xi, x_t^{(a)}), x_t^{(b)})}) \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}} - \nabla_a(f \circ b)|_{(m(\xi, x_t^{(a)}), x_t^{(b)})} \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}},$$

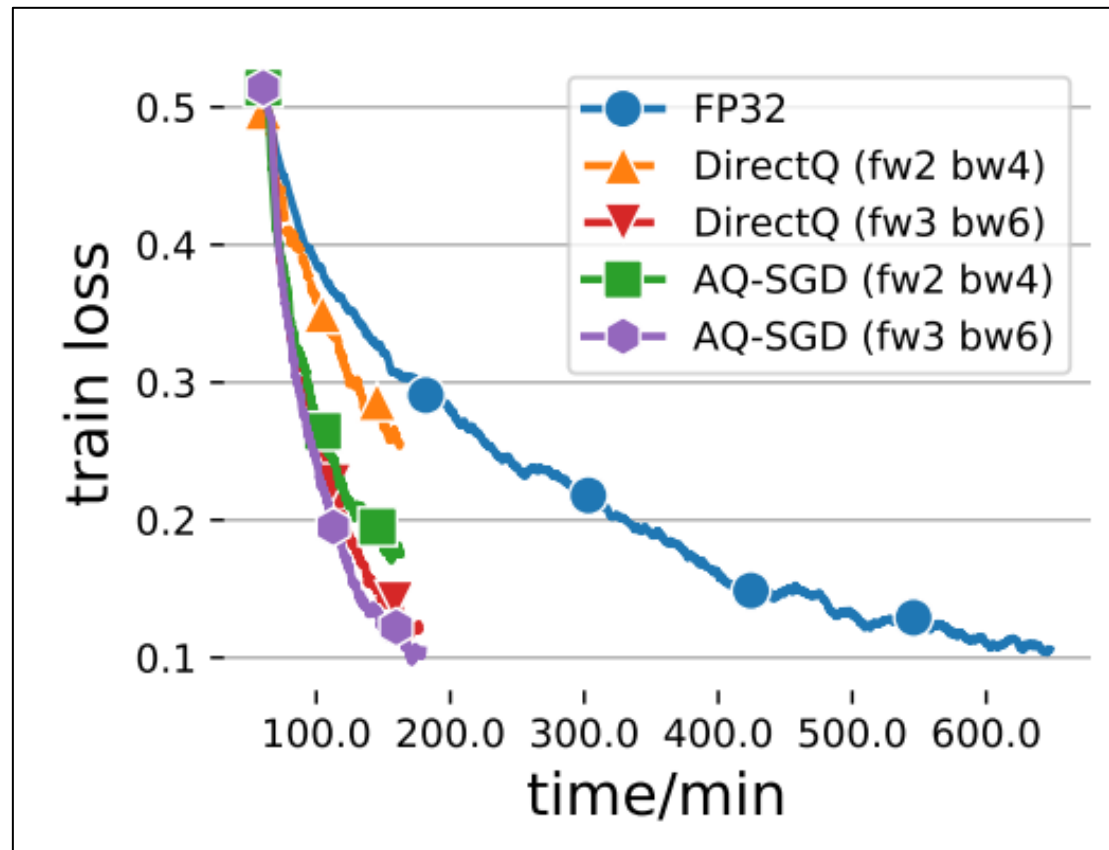
$$\Delta_\xi^{(a)}(x_t) = \nabla_a(f \circ b)|_{(m(\xi, x_t^{(a)}), x_t^{(b)})} \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}} - \nabla_a(f \circ b \circ a)|_{(x_t^{(a)}, x_t^{(b)})}$$

$$\Delta_\xi^{(b)}(x_t) = \nabla_{x^{(b)}}(f \circ b)|_{(m(\xi, x_t^{(a)}), x_t^{(b)})} - \nabla_{x^{(b)}}(f \circ b)|_{(a(\xi, x_t^{(a)}), x_t^{(b)})},$$

[1] J. Wang, et. al., “Fine-tuning Language Models over Slow Networks using Activation Quantization with Guarantees”, NeurIPS 2022

Communication compression in pipeline parallelism

Significant time saving



Perhaps too much theory? Need more practitioner

- 90% of distributed machine learning algorithms focus on convergence theories
- More research is needed to carefully evaluate communication-saving optimizers in real LLM training and finetuning tasks
- Transformers are standard models and Adam are standard optimizers in LLM
- We need to carefully design and evaluate communication-efficient optimizers for Transformer + Adam
- Communication-efficient optimizers for general non-convex problems have less practical values

Learn to distributed optimization

- Decentralized SGD
- Compressed SGD
- Local SGD or federated averaging

All these algorithms are designed by hands. They are not designed automatically.

Are these algorithms optimal? Probably not. Are there better algorithms? Probably yes.

How to design better algorithms? We do not know yet.

- Learn-to-learn: Replace hand-designed update rules with a learned update rule
- Classical gradient-based algorithm

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t).$$

- Learned gradient-based algorithm

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi).$$

g_t is a mapping to be learned. It is parameterized by ϕ . When taking a gradient, it outputs a better update direction

[1] M. Andrychowicz et. al., “Learning to learn by gradient descent by gradient descent”, NIPS 2016

Learned optimizer

- We can use RNN to learn a “good” update direction per iteration
- For each update, RNN takes in a gradient $\nabla f(\theta_t)$, outputs a learned update direction g_t
- Then the optimizer updates θ

$$h_t = p(U \nabla f(\theta_t) + V h_{t-1})$$

$$g_t = q(W h_t)$$

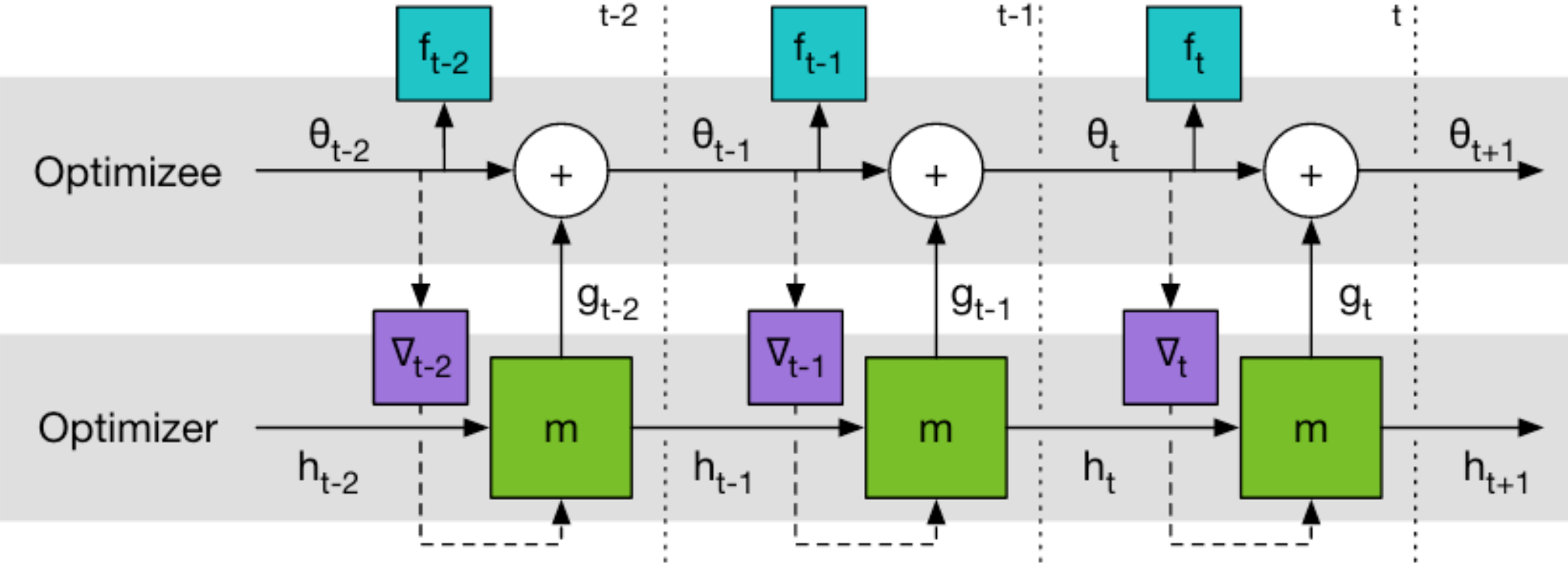
$$\theta_{t+1} = \theta_t + g_t$$

RNN; U, V, W are to be learned

Optimizer

[1] M. Andrychowicz et. al., “Learning to learn by gradient descent by gradient descent”, NIPS 2016

Learned optimizer



Learn to distributed optimization

- Decentralized SGD
- Compressed SGD
- Local SGD or federated averaging

Can we learn better distributed algorithms than those discussed in this lecture?

Can we learn better decentralized topology, or communication compressors?

How to effectively combine optimization and learning to develop algorithms?

Robust distributed optimization

- Distributed training over thousands of GPUs is extremely challenging
- Two major challenges: **stability** and **scalability**
- Stability: very common that some GPU crashes during training LLMs
 - Meta OPT-175B: 175+ job restarts caused by hardware failures in 2 months
 - InternLM: Waste 41700 GPU hours due to training crashes (>80% are infrastructure failures)
- Stability in LLM training is very important, we should develop algorithms that are tolerant with hardware failures

Thank you!

Kun Yuan homepage: <https://kunyuan827.github.io/>

We have openings for PostDocs

