

Don't Compress Gradients in Random Reshuffling: Compress Gradient Differences

Abdurakhmon Sadiev^{1,2}

Grigory Malinovsky¹

Eduard Gorbunov^{1,2,3,4}

Igor Sokolov¹

Ahmed Khaled⁵

Konstantin Burlachenko¹

Peter Richtárik¹

¹KAUST

²MIPT

³MBZUAI

⁴MILA

⁵Princeton University

The Problem

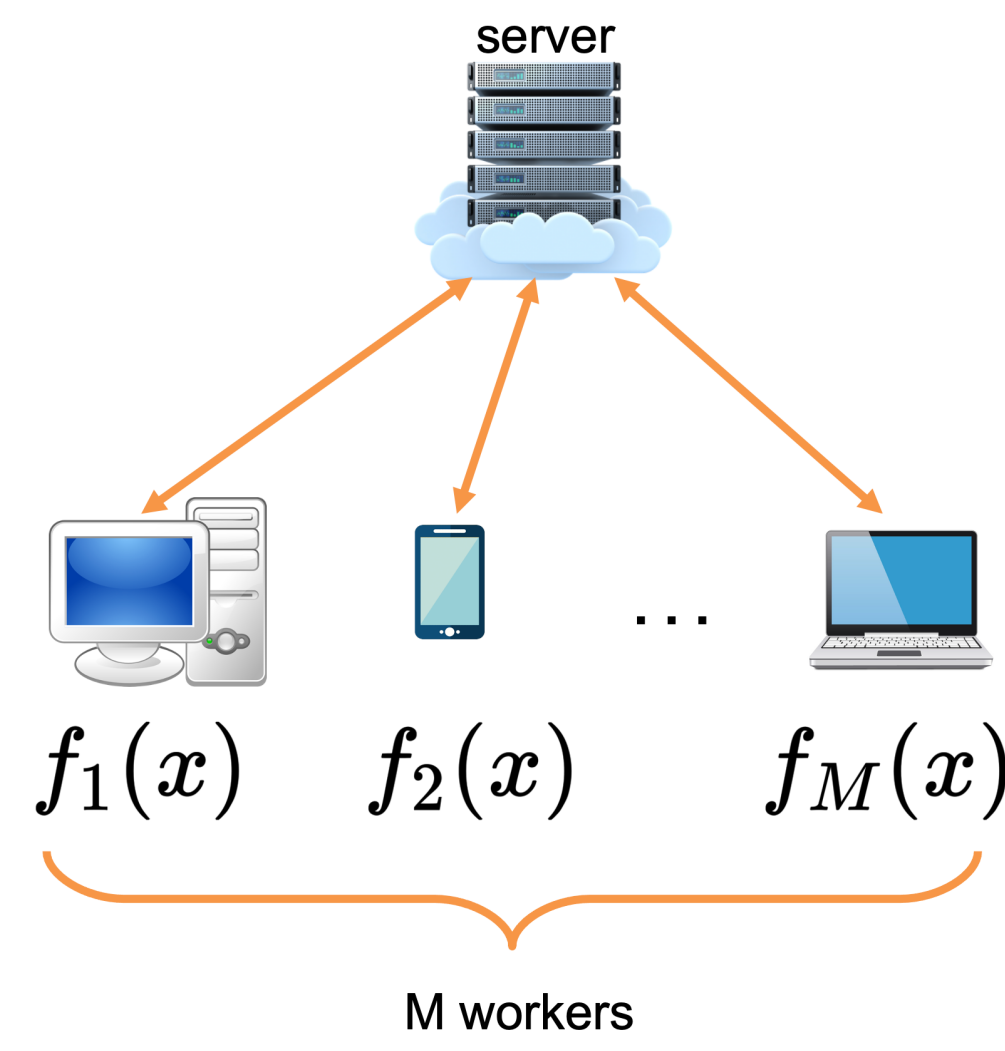
We study the optimization problem of Federated Learning (FL), which has the form

$$\min_{x \in \mathbb{R}^d} \left[f(x) := \frac{1}{M} \sum_{m=1}^M f_m(x) \right], \quad (1)$$

where M is the number of clients/devices and each function

$$f_m(x) := \frac{1}{n} \sum_{i=1}^n f_m^i(x), \quad (2)$$

represents the loss on client m .



Smoothness & Strong Convexity

- Smoothness

$$\|\nabla f_m^i(x) - \nabla f_m^i(y)\| \leq L_{i,m} \|x - y\|;$$

- Strong convexity of objective function

$$\frac{\mu}{2} \|x - y\|^2 \leq f(x) - f(y) - \langle \nabla f(y), x - y \rangle;$$

- Strong convexity of each losses on clients

$$\frac{\tilde{\mu}}{2} \|x - y\|^2 \leq f_m^i(x) - f_m^i(y) - \langle \nabla f_m^i(y), x - y \rangle;$$

for all $m \in [M]$ and $i \in [n]$.

References

- [1] Gorbunov, et al. "A Unified Theory of SGD: Variance Reduction, Sampling, Quantization and Coordinate Descent." AISTATS, 2020.
- [2] Reisizadeh, et al. "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization." AISTATS, 2020.
- [3] Malinovsky, et al. "Federated random reshuffling with compression and variance reduction." arXiv, 2022.
- [4] Haddadpour, et al. "Federated learning with compression: Unified analysis and sharp guarantees." AISTATS, 2021.

Compressed Learning

Unbiased Compressor

A compression operator is a randomized mapping $\mathcal{Q} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that for some $\omega > 0$

$$\mathbb{E}[\mathcal{Q}(x)] = x, \quad \mathbb{E}[\|\mathcal{Q}(x) - x\|^2] \leq \omega \|x\|^2$$

for all $x \in \mathbb{R}^d$.

- Rand- K sparsification operator is defined via

$$\mathcal{Q}(x) := \frac{d}{k} \sum_{i \in S} x_i e_i,$$

where $S \subseteq [d]$ is a subset of $[d]$ of cardinality k chosen uniformly at random. This is unbiased compressor with $\omega := \frac{d}{k} - 1$.

Main Goal

Design and analyze communication-efficient algorithms for Federated Learning using compression, random reshuffling, and/or local steps and improving upon existing algorithms both theoretically and practically.

Notation

- $[M] := \{1, \dots, M\}$

- Bregman divergence:

$$D_h(x, y) := h(x) - h(y) - \langle \nabla h(y), x - y \rangle;$$

- Maximal smoothness constant:

$$L_{\max} := \max_{i,m} L_{i,m};$$

- Iterate sequence $\{x_\star^i\}_i$:

$$x_\star^{i+1} = x_\star^i - \frac{\gamma}{M} \sum_{m=1}^M \nabla f_m^{\pi_m^i}(x_\star)$$

- Shuffling radius:

$$\sigma_{rad}^2 := \max_i \left\{ \frac{1}{\gamma^2 M} \sum_{m=1}^M \mathbb{E} D_{f_m^{\pi^i}}(x_\star^i, x_\star) \right\}$$

- Heterogeneity constant:

$$\zeta_\star^2 := \frac{1}{M} \sum_{m=1}^M \|\nabla f_m(x_\star)\|^2;$$

- Variances at the solution point x_\star :

$$\sigma_\star^2 := \frac{1}{Mn} \sum_{m=1}^M \sum_{i=1}^n \left\| \nabla f_m^i(x_\star) - \nabla f_m(x_\star) \right\|^2.$$

Q-RR

- 1: **Input:** x_0 – starting point, $\gamma > 0$ – stepsize
- 2: **for** $t = 0, 1, \dots, T-1$ **do**
- 3: Receive x_t from the server
- 4: $x_{t,m}^0 = x_t$
- 5: Sample random permutation of $[n]$:
 $\pi_m = (\pi_m^0, \dots, \pi_m^{n-1})$
- 6: **for** $i = 0, 1, \dots, n-1$ **do**
- 7: **for** $m = 1, \dots, M$ in parallel **do**
- 8: Receive x_t^i from the server
- 9: Compute and send $\mathcal{Q}(\nabla f_m^{\pi_m^i}(x_t^i))$
- 10: $x_t^{i+1} = x_t^i - \gamma \frac{1}{M} \sum_{m=1}^M \mathcal{Q}(\nabla f_m^{\pi_m^i}(x_t^i))$
- 11: Send x_t^{i+1} to the workers
- 12: $x_{t+1} = x_t^n$
- 13: **Output:** x_T

Q-RR [NEW]:

$$\tilde{\mathcal{O}} \left(\left(1 + \frac{\omega}{M} \right) \frac{L_{\max}}{\tilde{\mu}} + \frac{\omega (\zeta_\star^2 + \sigma_\star^2)}{M \tilde{\mu}^2 \varepsilon} + \frac{\sigma_{rad}}{\sqrt{\tilde{\mu}^3 \varepsilon}} \right)$$

QSGD [1]:

$$\tilde{\mathcal{O}} \left(\left(1 + \frac{\omega}{M} \right) \frac{L_{\max}}{\mu} + \frac{(\omega \zeta_\star^2 + (1 + \omega) \sigma_\star^2)}{M \mu^2 \varepsilon} \right).$$

- ✓ Easy to implement;

- ✓ Memory friendly (does not require storing any additional vectors).

✗ Q-RR has no theoretical advantages over QSGD unless ω is very small. This phenomenon is also observed in the experiments.

This method illustrates non-triviality of proper generalization of RR to the distributed learning with compression.

DIANA-RR

- 1: **Input:** x_0 – starting point, $\{h_{0,m}^i\}_{m,i=1,1}^{M,n}$ – initial shift-vectors, $\gamma > 0$ – stepsize, $\alpha > 0$ – stepsize for learning the shifts
- 2: **for** $t = 0, 1, \dots, T-1$ **do**
- 3: Receive x_t from the server
- 4: $x_{t,m}^0 = x_t$
- 5: Sample random permutation of $[n]$:
 $\pi_m = (\pi_m^0, \dots, \pi_m^{n-1})$
- 6: **for** $i = 0, 1, \dots, n-1$ **do**
- 7: **for** $m = 1, 2, \dots, M$ in parallel **do**
- 8: Receive x_t^i from the server
- 9: Compute and send $\mathcal{Q}(\nabla f_m^{\pi_m^i}(x_t^i) - h_{t,m}^{\pi_m^i})$
- 10: $\hat{g}_{t,m}^{\pi_m^i} = h_{t,m}^{\pi_m^i} + \mathcal{Q}(\nabla f_m^{\pi_m^i}(x_{t,m}^i) - h_{t,m}^{\pi_m^i})$
- 11: $h_{t+1,m}^{\pi_m^i} = h_{t,m}^{\pi_m^i} + \alpha \mathcal{Q}(\nabla f_m^{\pi_m^i}(x_{t,m}^i) - h_{t,m}^{\pi_m^i})$
- 12: $x_t^{i+1} = x_t^i - \gamma \frac{1}{M} \sum_{m=1}^M \hat{g}_{t,m}^{\pi_m^i}$
- 13: Send x_t^{i+1} to the workers
- 14: $x_{t+1} = x_t^n$
- 15: **Output:** x_T

DIANA-RR [NEW]:

$$\tilde{\mathcal{O}} \left(n(1 + \omega) + \left(1 + \frac{\omega}{M} \right) \frac{L_{\max}}{\tilde{\mu}} + \frac{\sigma_{rad}}{\sqrt{\varepsilon \tilde{\mu}^3}} \right)$$

DIANA [1]:

$$\tilde{\mathcal{O}} \left(\left(1 + \frac{\omega}{M} \right) \frac{L_{\max}}{\mu} + \frac{(1 + \omega) \sigma_\star^2}{M \mu^2 \varepsilon} \right).$$

- ✓ Unlike Q-RR, DIANA-RR does not have a $\tilde{\mathcal{O}}(1/\varepsilon)$ term;

- ✓ Overall complexity of DIANA-RR improves over DIANA, since $\mathcal{O}(\sigma_{rad}/\sqrt{\varepsilon \tilde{\mu}^3})$ has a better dependence on ε than $\mathcal{O}((1 + \omega) \sigma_\star^2 / (M \mu^2 \varepsilon))$.

- ✗ It can be memory expensive to maintain $\{h_{t,m}^i\}_{m \in [M], i \in [n]}$ shifts.

Experiments

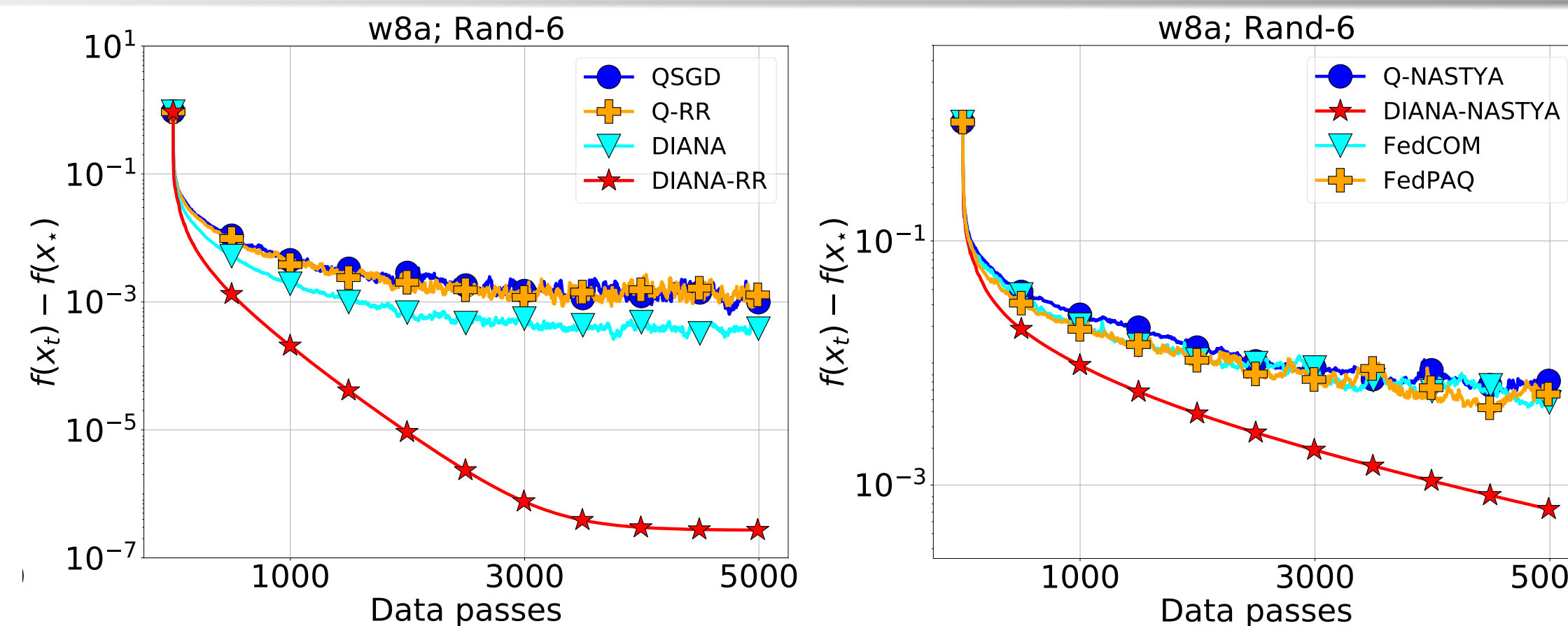


Figure 1: The comparison of Q-NASTYA, DIANA-NASTYA, Q-RR, DIANA-RR and existing baselines (FedCOM, FedPAQ) on **binary classification problem** with $M = 10$ workers. Stepsizes were tuned and workers used Rand-k compressor with $k/d = 0.02$ ($k = 6, d = 300$).

Q-NASTYA

- 1: **Input:** x_0 – starting point, $\gamma > 0$ – local stepsize, $\eta > 0$ – global stepsize
- 2: **for** $t = 0, 1, \dots, T-1$ **do**
- 3: **for** $m = 1, \dots, M$ in parallel **do**
- 4: Receive x_t from the server
- 5: $x_{t,m}^0 = x_t$
- 6: Sample random permutation of $[n]$:
 $\pi_m = (\pi_m^0, \dots, \pi_m^{n-1})$
- 7: **for** $i = 0, 1, \dots, n-1$ **do**
- 8: $x_{t,m}^{i+1} = x_{t,m}^i - \gamma \nabla f_m^{\pi_m^i}(x_{t,m}^i)$
- 9: $g_{t,m} = \frac{1}{\gamma n} (x_t - x_{t,m}^n)$
- 10: Send $\mathcal{Q}_t(g_{t,m})$ to the server
- 11: $g_t = \frac{1}{M} \sum_{m=1}^M \mathcal{Q}_t(g_{t,m})$
- 12: $x_{t+1} = x_t - \eta g_t$
- 13: Send x_{t+1} to the workers
- 14: $x_T = x_T^n$
- 15: **Output:** x_T

Q-NASTYA [NEW]:

$$\tilde{\mathcal{O}} \left(\frac{L_{\max}}{\mu} \left(1 + \frac{\omega}{M} \right) + \frac{\omega \zeta_\star^2}{M \varepsilon \mu^3} + \sqrt{\frac{L_{\max}}{\varepsilon \mu^3}} \sqrt{\zeta_\star^2 + \frac{\sigma_\star^2}{n}} \right)$$

FedPAQ [2]:

$$\tilde{\mathcal{O}} \left(\frac{L_{\max}}{\mu} \left(1 + \frac{\omega}{M} \right) + \frac{\omega \sigma^2}{M \mu^2 \varepsilon} + \frac{\sigma^2}{M \mu^2 \varepsilon} \right).$$

- ✓ Unlike FedCOM [4], Q-NASTYA provably works in a fully heterogeneous regime;

- ✓ Unlike FedPAQ, analysis of Q-NASTYA does not rely on the bounded variance assumption;

- ✓ Unlike FedCRR [3], Q-NASTYA converges for any $\omega \geq 0$;

- ✓ If ω is small, complexity of Q-NASTYA is superior to FedPAQ.

- ✗ In the big ω regime, Q-NASTYA has the same $\tilde{\mathcal{O}}(1/\varepsilon)$ dependence as FedPAQ.

DIANA-NASTYA

- 1: **Input:** x_0 – starting point, $\{h_{0,m}\}_{m=1}^M$ – initial shift-vectors, $\gamma > 0$ – local stepsize, $\eta > 0$ – global stepsize, $\alpha > 0$ – stepsize for learning the shifts
- 2: **for** $t = 0, 1, \dots, T-1$ **do**
- 3: **for** $m = 1, \dots, M$ in parallel **do**
- 4: Receive x_t from the server
- 5: $x_{t,m}^0 = x_t$
- 6: Sample random permutation of $[n]$:
 $\pi_m = (\pi_m^0, \dots, \pi_m^{n-1})$
- 7: **for** $i = 0, 1, \dots, n-1$ **do**
- 8: $x_{t,m}^{i+1} = x_{t,m}^i - \gamma \nabla f_m^{\pi_m^i}(x_{t,m}^i)$
- 9: $g_{t,m} = \frac{1}{\gamma n} (x_t - x_{t,m}^n)$
- 10: Send $\mathcal{Q}_t(g_{t,m} - h_{t,m})$ to the server
- 11: $h_{t+1,m} = h_{t,m} + \alpha \mathcal{Q}_t(g_{t,m} - h_{t,m})$
- 12: $\hat{g}_{t,m} = h_{t,m} + \mathcal{Q}_t(g_{t,m} - h_{t,m})$
- 13: $h_{t+1} = h_t + \frac{\alpha}{M} \sum_{m=1}^M \mathcal{Q}_t(g_{t,m} - h_{t,m})$
- 14: $\hat{g}_t = h_t + \frac{1}{M} \sum_{m=1}^M \mathcal{Q}_t(g_{t,m} - h_{t,m})$
- 15: $x_{t+1} = x_t - \eta \hat{g}_t$
- 16: **Output:** x_T

DIANA-NASTYA [NEW]:

$$\tilde{\mathcal{O}} \left(\omega + \frac{L_{\max}}{\mu} \left(1 + \frac{\omega}{M} \right) + \sqrt{\frac{L_{\max}}{\varepsilon \mu^3}} \sqrt{\zeta_\star^2 + \frac{\sigma_\star^2}{n}} \right)$$

FedCRR-VR [3]:

$$\tilde{\mathcal{O}} \left(\frac{(\omega + 1) \left(1 - \frac{1}{\kappa} \right)^n}{\left(1 - \left(1 - \frac{1}{\kappa} \right)^n \right)^2} + \frac{\sqrt{\kappa} (\zeta_\star + \sigma_\star)}{\mu \sqrt{\varepsilon}} \right)$$

- ✓ The complexity of DIANA-NASTYA is superior to both FedPAQ and Q-NASTYA;

- ✓ If $\kappa := \frac{L_{\max}}{\mu} \gg 1$, complexity of DIANA-NASTYA is better than for FedCRR-VR.

- Each worker i has to maintain an additional vector state $h_{t,m}$, which causes an additional memory cost.

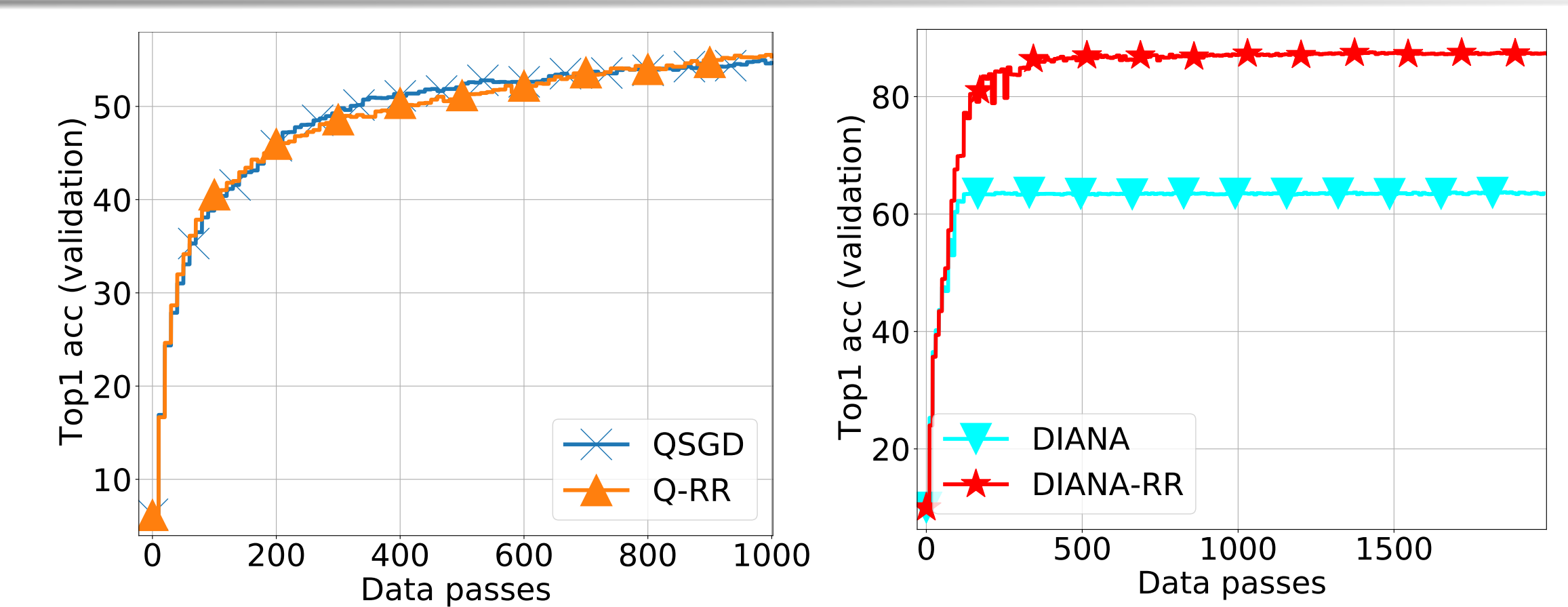


Figure 2: The comparison of Q-RR, QSGD, DIANA, and DIANA-RR on the task of training ResNet-18 on CIFAR-10 with $M = 10$ workers. Stepsizes were tuned and workers used Rand-k compressor with $k/d = 0.05$.