Federated Learning: Strategies for Improving Communication Efficiency

Jakub Konečný^, H. Brendan McMahan*, Felix X Yu*, Peter Richtárik^, Ananda Theertha Suresh*, Dave Bacon*

^School of Mathematics, University of Edinburgh, *Google, Inc.

Federated Learning: machine learning setting where the goal is to train a high-quality centralized model with training data distributed over a large number of clients (e.g. phones), each with unreliable and relatively slow network connections.

A prototypical round consists of:

1. Select some clients, each downloads current model
2. Each client updates the model, based on local data
3. The updates are uploaded back to server
4. Server aggregates the updates (e.g. by summing), and forms an improved global model

Model state at time t (downloaded by clients)

Update by client i, uploaded to server (based on W_t and local data)

$$W_{t+1} = W_t + \eta_t H_t, \qquad H_t = \sum_i H_t^i$$

Improved global model

Aggregated updates (can be more complex)

Reasons why Step 3 can be a practical bottleneck:

- Asymmetric internet connections – slower upload
- Additional cryptographic protocols for privacy reasons – expansion in # of bits communicated

Goal:
Reduce the size of updates H_t^i uploaded to server, in bits, without sacrificing (much of) the performance

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. "Federated Learning: Strategies for Improving Communication Efficiency." *arXiv:1610.05492* (2016).
[2] H. B. McMahan, E. Moore, D. Ramage, and B. Aguera y Arcas. "Federated Learning of Deep Networks using Model Averaging." arXiv:1602.05629 (2016).

Structured Updates: Enforce client update $H_t^i \in \mathbb{R}^{d_1 \times d_2}$ to be of pre-specified structure.

- **Low Rank:** Express $H_t^i = A_t^i B_t^i$, where

  $$A_t^i \in \mathbb{R}^{d_1 \times k} \qquad B_t^i \in \mathbb{R}^{k \times d_2}$$

  Randomly initialized (kept fixed throughout)

  Train only this matrix (and communicate)

  A_t^i can be compressed as a random seed
- **Random Mask:** Enforce the update H_t^i to be a sparse matrix, with a pre-defined random sparsity pattern, communicating only its non-zero values.

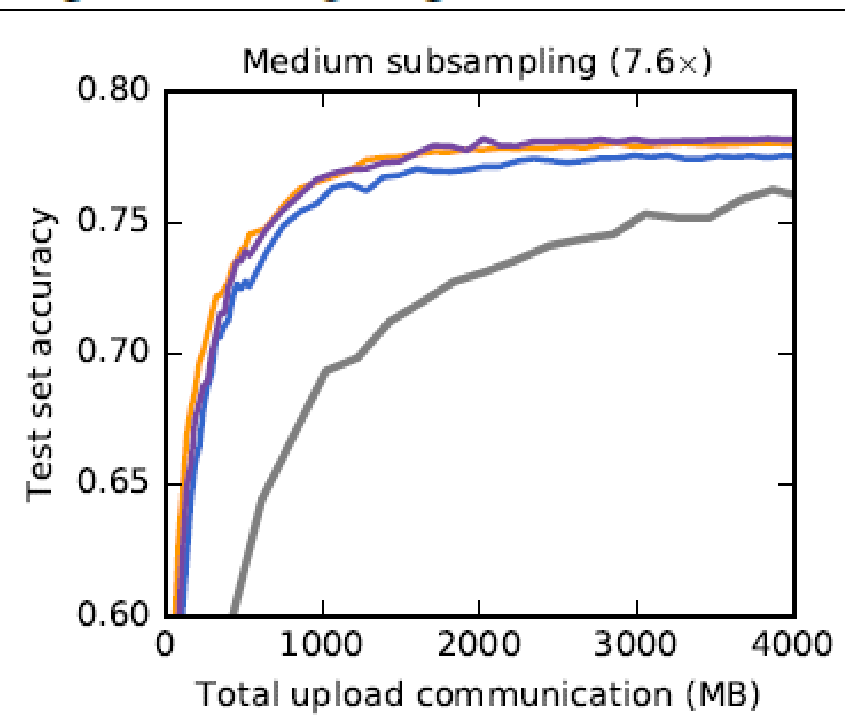
Sketched Updates: Train update H_t^i without constraints, encode in a (lossy) compressed form and send to server, using one or more of the following tools combined.

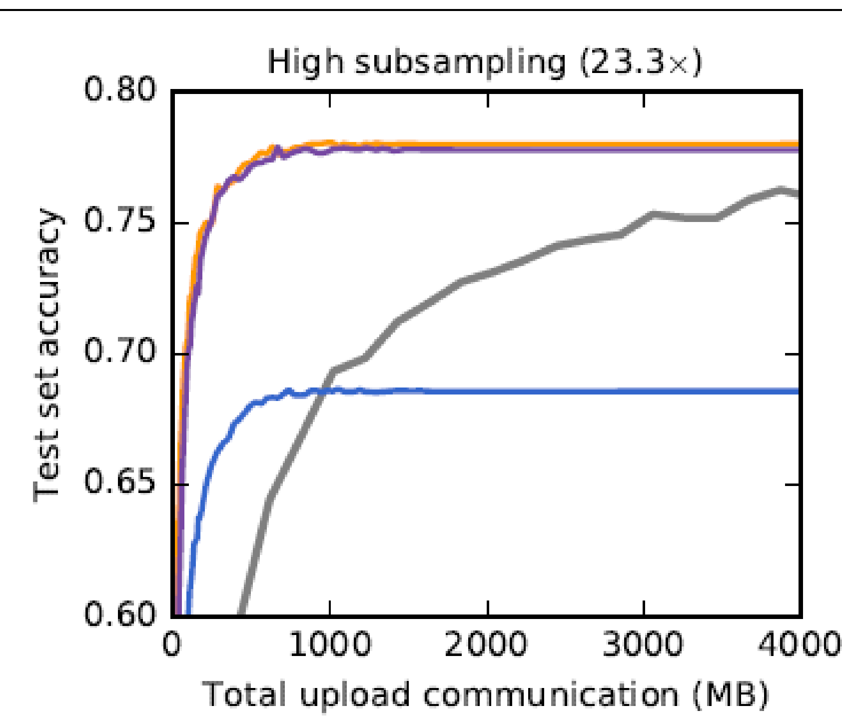
- **Random Mask:** Randomly subsample and scale the update on a per-element basis.
- **Binary Quantization:** Consider $h_{\max}, h_{\min}$ to be the largest and smallest elements of H_t^i.
  We quantize every element h of H_t^i as follows

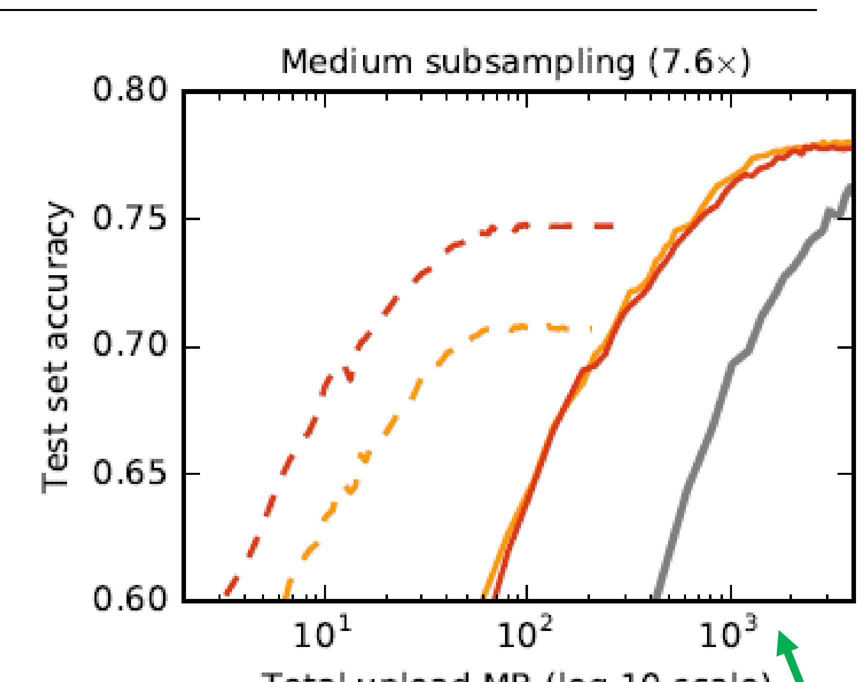
  $$\tilde{h} = \begin{cases} h_{\max}, & \text{with probability } \frac{h - h_{\min}}{h_{\max} - h_{\min}} \\ h_{\min}, & \text{with probability } \frac{h_{\max} - h}{h_{\max} - h_{\min}} \end{cases}$$

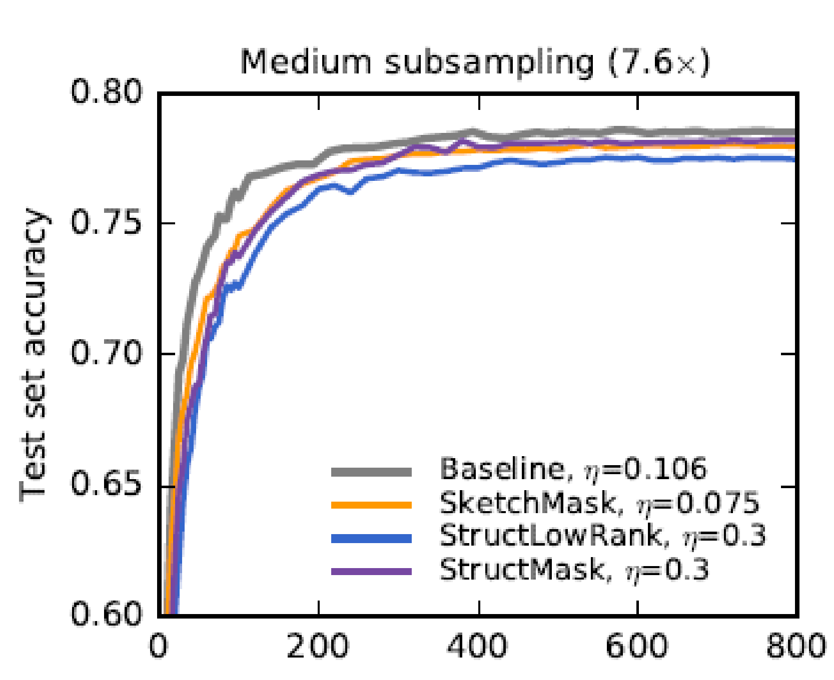
  Can be verified this yields an unbiased estimate.
- **Random Structured Rotation:** Generate ($\mathcal{O}(d)$) and apply ($\mathcal{O}(d \log d)$) randomized structured rotation as preprocessing, apply inverse rotation on server. Randomness compressed in random seed.

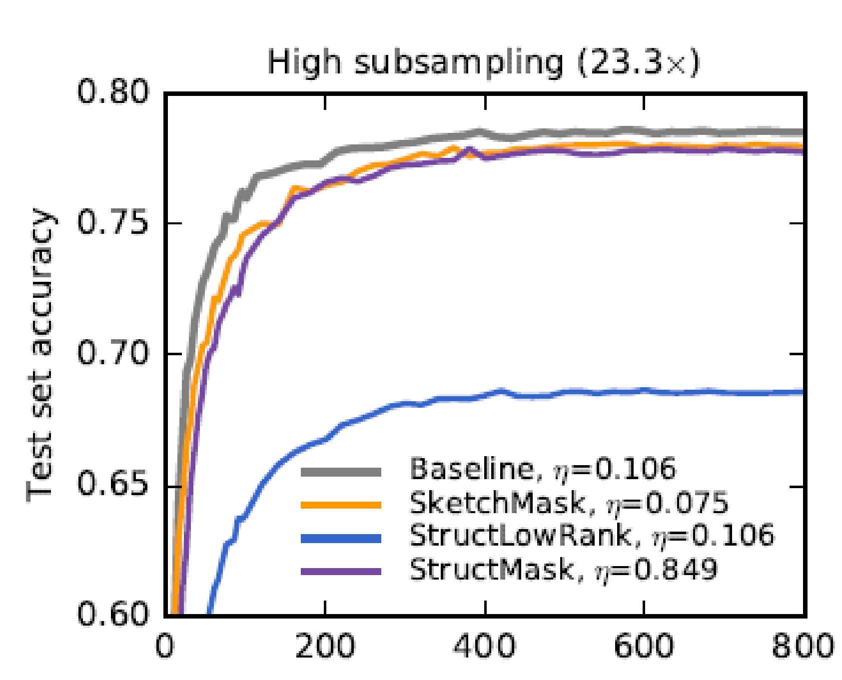
	(Low) Rank	Sampling Probabilities	model size	reduction
Full Model (baseline)	64, 64, 384, 192	1, 1, 1, 1	4.075 MB	—
Medium subsampling	64, 64, 12, 6	1, 1, 0.03125, 0.03125	0.533 MB	7.6×
High subsampling	8, 8, 12, 6	0.125, 0.125, 0.03125, 0.03125	0.175 MB	23.3×

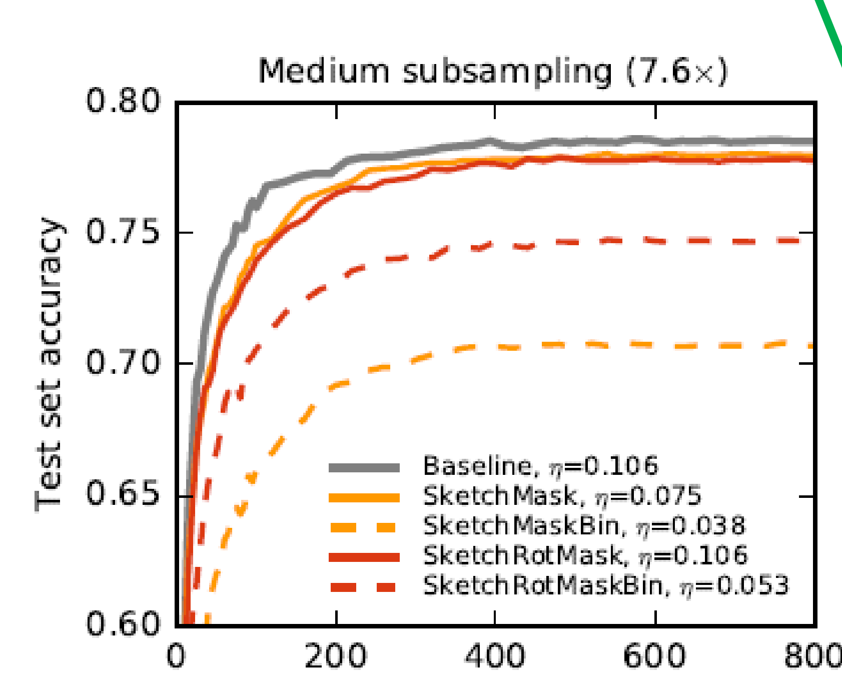












Experiments: [1]

- CIFAR data
- Partitioned (randomly) to 100 clients, each with 500 data points
- TF tutorial model
- Base optimizer [2] Model Averaging

Major improvement in terms of total MB uploaded vs convergence speed

Quantization with masking can train model while in total communicating much less than the size of original data.