Client Selection and Model Compression

Nícolas Riccieri Gardin Assumpção

Basic Protocol of Federated Learning

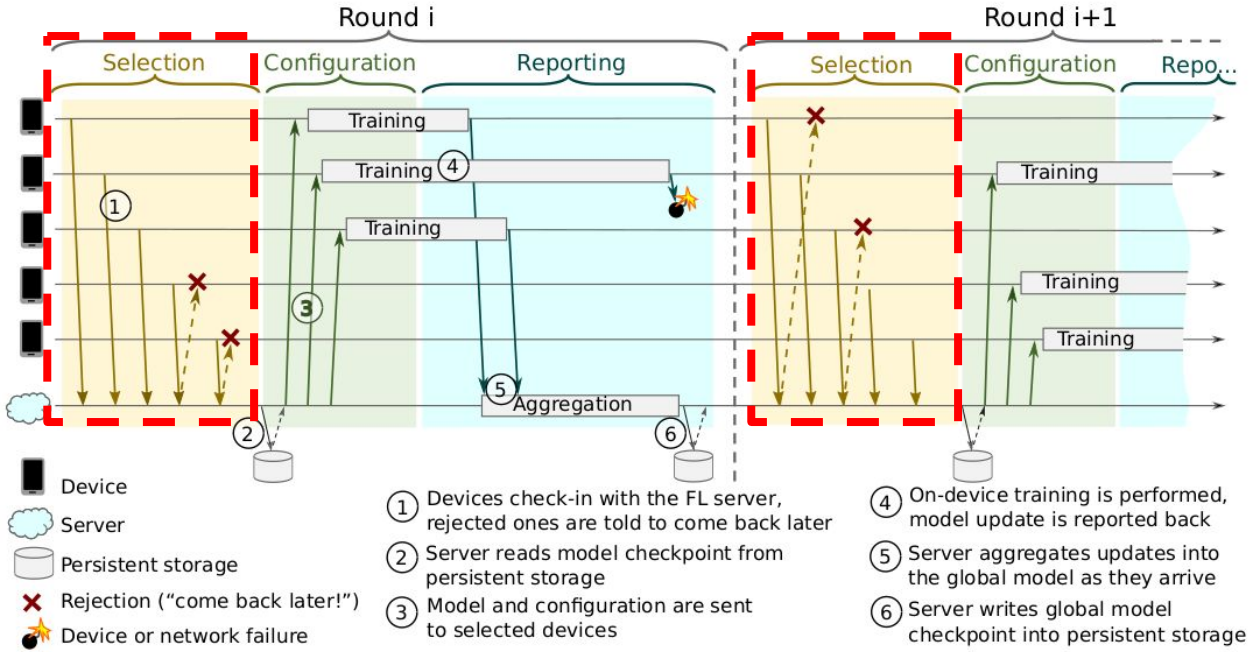


Figure 1: Federated Learning Protocol

(Bonawitz et al. 2019)

Power of Choice

CHO, Yae Jee; **WANG**, Jianyu; **JOSHI**, Gauri. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv preprint arXiv:2010.01243, 2020.

Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies

Yae Jee Cho
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213
yaejeec@andrew.cmu.edu

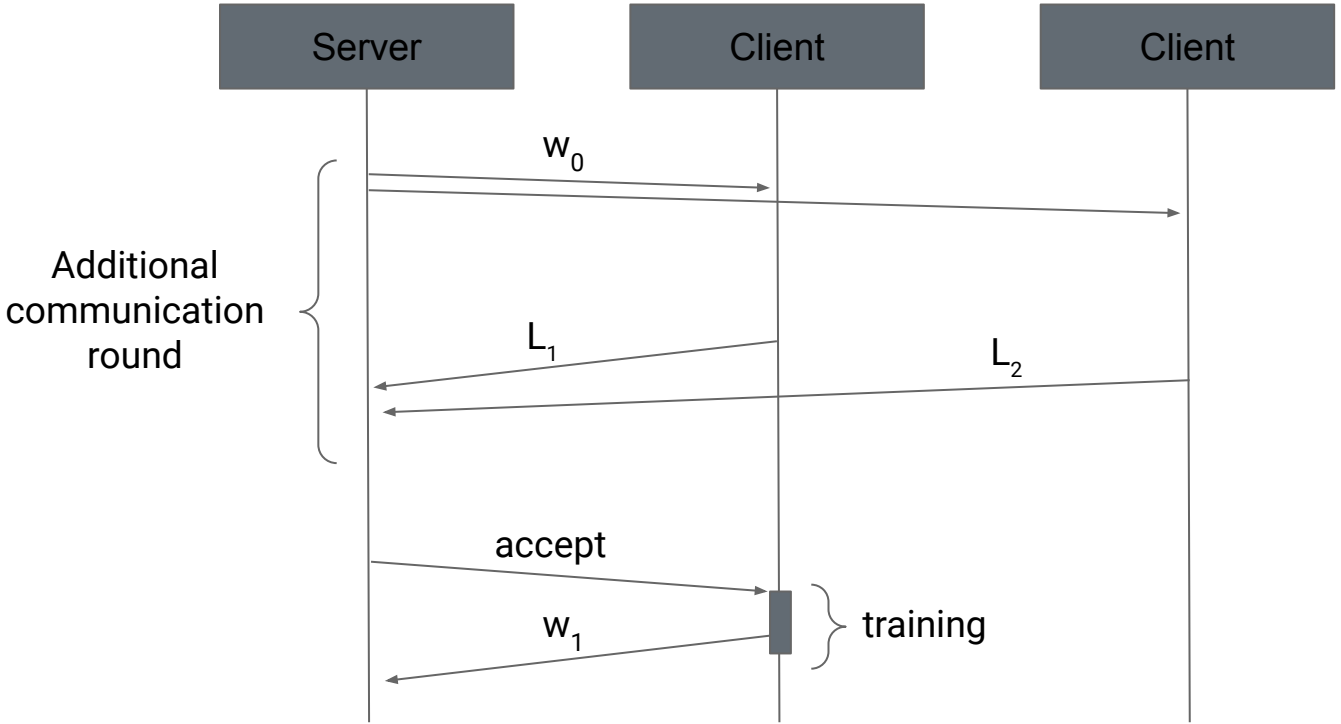
Jianyu Wang
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213
jianyuw1@andrew.cmu.edu

Gauri Joshi
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213
gaurij@andrew.cmu.edu

Abstract

Federated learning is a distributed optimization paradigm that enables a large number of resource-limited client nodes to cooperatively train a model without data sharing. Several works have analyzed the convergence of federated learning by accounting of data heterogeneity, communication and computation limitations, and partial client participation. However, they assume unbiased client participation, where clients are selected at random or in proportion of their data sizes. In this paper, we present the first convergence analysis of federated optimization for biased client selection strategies, and quantify how the selection bias affects convergence speed. We reveal that biasing client selection towards clients with higher local loss achieves faster error convergence. Using this insight, we propose POWER-OF-CHOICE, a communication- and computation-efficient client selection framework that can flexibly span the trade-off between convergence speed and solution bias. Our experiments demonstrate that POWER-OF-CHOICE strategies converge up to 3× faster and give 10% higher test accuracy than the baseline random selection.

Client selection in federated learning: Convergence analysis and power-of-choice selection strategies



Client selection in federated learning: Convergence analysis and power-of-choice selection strategies

Communication efficient version:

1. The server initiate the loss estimation of each client as ∞
2. The server selects N random clients
3. The server chose K<N clients with the lowest loss value
4. The server sends the model's parameters to the K selected clients
5. The clients train the model using their local data.
6. The clients send the updated models back to the server with the loss value.
7. The server aggregate the models and update the loss estimation of the K clients.
8. Repeat from step 2.

Count Sketch

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., ... & Arora, R. (2020, November). Fetchsgd: Communication-efficient federated learning with sketching. In International Conference on Machine Learning (pp. 8253-8265). PMLR.

Algorithm 1 `FetchSGD`

Input: number of model weights to update each round k
Input: learning rate η
Input: number of timesteps T
Input: momentum parameter ρ, local batch size ℓ
Input: Number of clients selected per round W
Input: Sketching and unsketching functions $\mathcal{S}, \mathcal{U}$

Initialize S_u^0 and S_e^0 to zero sketches
Initialize w^0 using the same random seed on the clients and aggregator
for $t = 1, 2, \cdots T$ **do**
  Randomly select W clients $c_1, \ldots c_W$
  loop {In parallel on clients $\{c_i\}_{i=1}^W$}
    Download (possibly sparse) new model weights $w^t - w^0$
    Compute stochastic gradient g_i^t on batch B_i of size ℓ: $g_i^t = \frac{1}{\ell} \sum_{j=1}^{l} \nabla_w \mathcal{L}(w^t, z_j)$
    Sketch g_i^t: $S_i^t = \mathcal{S}(g_i^t)$ and send it to the Aggregator
  end loop
  Aggregate sketches $S^t = \frac{1}{W} \sum_{i=1}^{W} S_i^t$
  Momentum: $S_u^t = \rho S_u^{t-1} + S^t$
  Error feedback: $S_e^t = \eta S_u^t + S_e^t$
  Unsketch: $\Delta^t = \text{Top-k}(\mathcal{U}(S_e^t))$
  Error accumulation: $S_e^{t+1} = S_e^t - S(\Delta^t)$
  Update $w^{t+1} = w^t - \Delta^t$
end for

Output: $\{w^t\}_{t=1}^T$

Definition

Let's compress the vector $V = [v_1, v_2, \ldots, v_n]^T$ using Count Sketch.

To do so, we will use a table C with R rows and C columns.

We choose 2R pairwise independent hash functions:
$h_1, h_2, \ldots, h_R : [n] \rightarrow [C]$
$s_1, s_2, \ldots, s_R : [n] \rightarrow \{-1, +1\}$

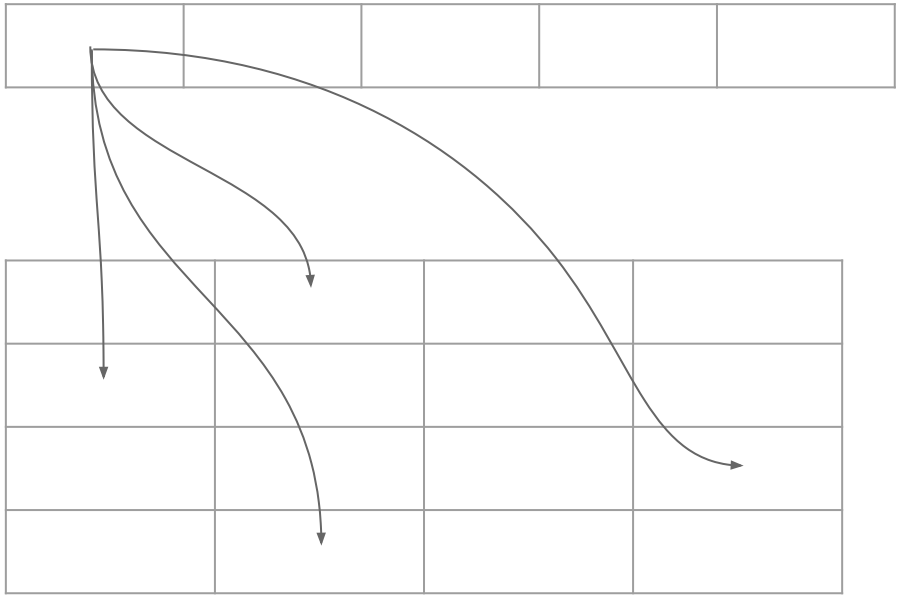
The elements of table C is according:

$$y_j^{(i)} = \sum_{u:h_i(u)=j} x_u s_i(u).$$

The algorithm will then output the estimate:

$$\widehat{x}_u = \text{median}_{i \in [R]} y_{h_i(u)}^{(i)} s_i(u)$$

Illustration



Example – Compression

$h_1(x) = \mathrm{mod}_3\ x$
$h_2(x) = \mathrm{mod}_3\ 2x$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ x$

s =

+	+	+	-	-
-	+	-	-	+
-	-	+	+	+

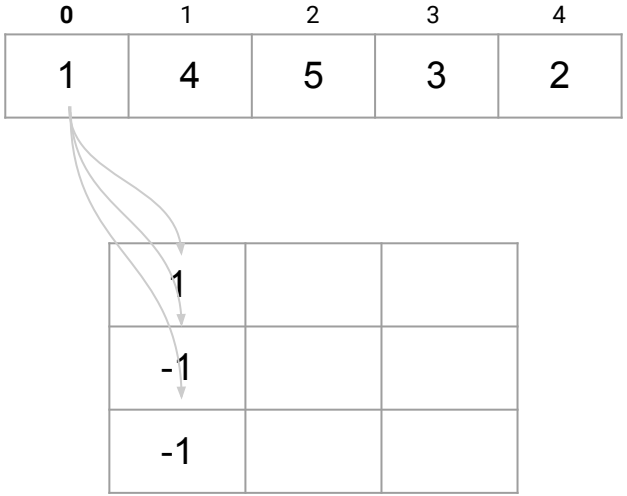
0	1	2	3	4
1	4	5	3	2

Example – Compression

$h_1(x) = \mathrm{mod}_3\ 0 = 0$
$h_2(x) = \mathrm{mod}_3\ 2 \cdot 0 = 0$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ 0 = 0$

s =

+	+	+	-	-
-	+	-	-	+
-	-	+	+	+

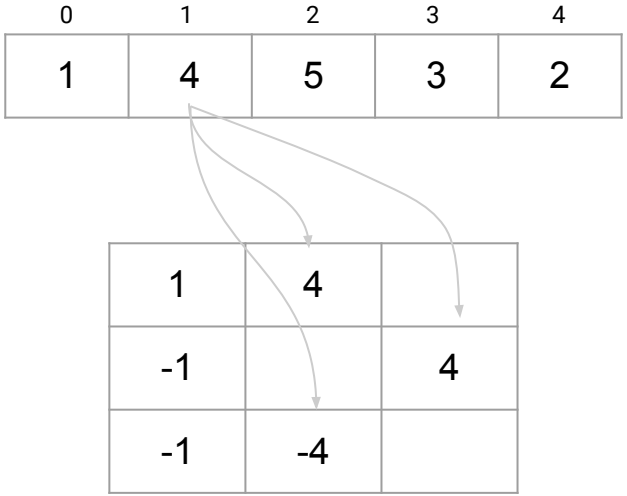


Example – Compression

$h_1(x) = \mathrm{mod}_3\ 1 = 1$
$h_2(x) = \mathrm{mod}_3\ 2 \cdot 1 = 2$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ 1 = 1$

s =

+	**+**	+	-	-
-	**+**	-	-	+
-	**-**	+	+	+



Example – Compression

$h_1(x) = \mathrm{mod}_3\ x$
$h_2(x) = \mathrm{mod}_3\ 2x$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ x$

s =

+	+	+	-	-
-	+	-	-	+
-	-	+	+	+

0	1	2	3	4
1	4	5	3	2

-2	2	5
-4	-5	6
4	-4	5

Example – Uncompression

$h_1(x) = \mathrm{mod}_3\ x$
$h_2(x) = \mathrm{mod}_3\ 2x$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ x$

-2	2	5
-4	-5	6
4	-4	5

s =

+	+	+	-	-
-	+	-	-	+
-	-	+	+	+

Median (-2, -4, 4) = -2

0	1	2	3	4
-2				

Example – Uncompression

$h_1(x) = \mathrm{mod}_3\ x$
$h_2(x) = \mathrm{mod}_3\ 2x$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ x$

s =	+	+	+	-	-
	-	+	-	-	+
	-	-	+	+	+

-2	2	5
-4	-5	6
4	-4	5

Median (2, 6, 4) = 4

0	1	2	3	4
-2	**4**			

Example – Uncompression

$h_1(x) = \mathrm{mod}_3\ x$
$h_2(x) = \mathrm{mod}_3\ 2x$
$h_3(x) = \mathrm{mod}_3\ \mathrm{mod}_4\ x$

s =	+	+	+	-	-
	-	+	-	-	+
	-	-	+	+	+

-2	2	5
-4	-5	6
4	-4	5

Median (5, 5, 5) = 5

0	1	2	3	4
-2	4	**5**		

Example

Original

Estimated

Properties

$$\text{CountSketch}(v_1) + \text{CountSketch}(v_2) = \text{CountSketch}(v_1 + v_2)$$

$$\text{TopK(estimated)} \approx \text{TopK(original)}$$

Count Sketch - Error Propagation

$$
\begin{aligned}
\mathrm{S}^t &= \frac{1}{W}\sum_{i=1}^{W}\mathcal{S}(\mathrm{g}_i^t) \\
\mathrm{S}_u^{t+1} &= \rho \mathrm{S}_u^t + \mathrm{S}^t \\
\Delta &= \text{Top-k}(\mathcal{U}(\eta \mathrm{S}_u^{t+1} + \mathrm{S}_e^t)) \\
\mathrm{S}_e^{t+1} &= \eta \mathrm{S}_u^{t+1} + \mathrm{S}_e^t - \mathcal{S}(\Delta) \\
\mathrm{w}^{t+1} &= \mathrm{w}^t - \Delta.
\end{aligned}
$$

Compressed Client Selection

Aissa Hadj **Mohamed**, Nícolas R. G. **Assumpção**, Carlos A. **Astudillo**, Allan M. de **Souza**, Luiz F. **Bittencourt** and Leandro A. **Villas**.

2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)

Compressed Client Selection for Efficient Communication in Federated Learning

Aissa Hadj Mohamed, Nícolas R. G. Assumpção, Carlos A. Astudillo, Allan M. de Souza,
Luiz F. Bittencourt, and Leandro A. Villas
Institute of Computing - State University of Campinas, Brazil
a265189@dac.unicamp.br, n121245@dac.unicamp.br, allanms@lrc.ic.unicamp.br, {castudillo,bit,leandro}@ic.unicamp.br

***Abstract*—Federated learning (FL) is a distributed approach that enables collaborative training of a shared machine learning (ML) model for a given task. FL requires a high level of communication between devices and a central server, which is a cause of many issues such as communication bottlenecks and scaling in the network. Therefore, we introduce the CCS (Compressed Client Selection) algorithm aimed at decreasing the overall communication costs for fitting a model in the FL environment. CCS employs a biased client selection strategy that reduces the number of devices training the ML model and the number of rounds required to reach convergence. In addition, the compression method Count Sketch is implemented to reduce the overhead in client-to-server communication. A use case on the Human Activity Recognition dataset is performed to evaluate CCS and compare it with other state-of-the-art approaches. Experimental evaluations show that CCS efficiently reduces the overall communication overhead for fitting a model and its convergence in a FL environment. In particular, CCS reduces up to 90% the communication overhead compared to literature approaches while providing good convergence even in scenarios where the data are not-independently and identically distributed among client devices.**

***Index Terms*—Federated Learning, Communication, Machine Learning**

I. Introduction

With the advent of technology, there has been an unprecedented generation of data by an increasing number of devices such as smartphones, sensors, and machines. Deep learning (DL) leverages this vast amount of data to build models for

a new communication cycle. This process is repeated until convergence.

In this FL process, there are many challenges related to the communication between the clients and the central server. Devices connected over the network have to constantly share their updates which can cause a communication bottleneck. Due to limited bandwidth, energy, and power resources of the client devices, the rounds of communication can also be slow [2]. Therefore, it is essential to make the communication in a FL environment as efficient as possible [3].

As mentioned by Shahid *et al.* [4], the following are some of the known challenges that could create a communication bottleneck in a FL environment:

Number of participating devices: a high number of FL clients results in more data for the global model to train on, which improves the prediction results or accuracy score. However, a significant number of devices participating in the training may also create a communication bottleneck [3].

Communication capacity: the participating devices may not always have an adequate or reliable communication link [5].

Limited power of client devices: the global model training relies on limited client device resources as opposed to powerful CPUs and GPUs in a classic centralized setting. In particular, clients have limited computation, power, and storage resources[6].

Statistical heterogeneity: the data of the clients participating

CCS – Compressed Client Selection

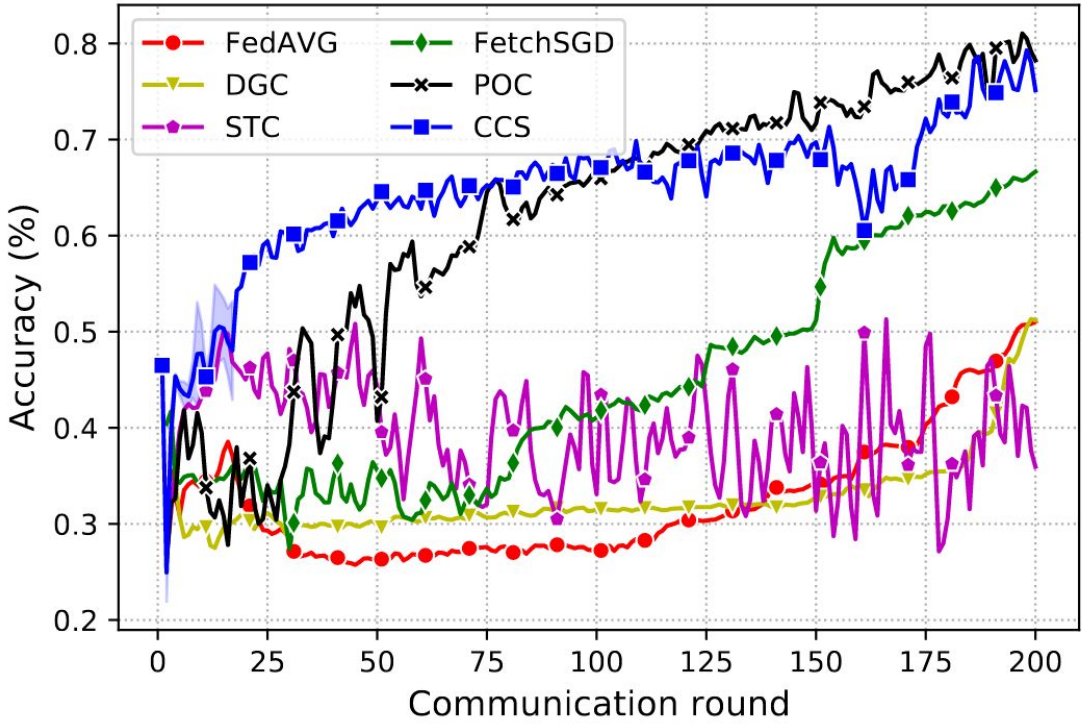
Central Server do:

$W \leftarrow RandomInitialization()$ ▷ *initialize the model*
$m \leftarrow max(N \cdot p, 1)$ ▷ *selects a set o clients randomly*
for Each communication round $1, 2, \ldots$ **do**
  $S \leftarrow clientSelection(N, m)$
  for client $n \in S$ **do** ▷ *in parallel*
    $w_{t+1}^{n} \leftarrow sendTrainRequest(d_n, w_t)$
  end for
  $w_{t+1}^{c} \leftarrow \sum_{n \in S} p_n w_{t+1}^{n}$ ▷ *aggregates the gradients received*
  $\Delta_t \leftarrow Unsketch(w_{t+1}^{c})$ ▷ *decompress the gradients received*
  $w_{t+1} \leftarrow w_t - \text{top-k}(\Delta_t)$ ▷ *update the global model*
  Broadcast(w_{t+1}, N) ▷ *central server broadcasts updated model to N clients*
end for

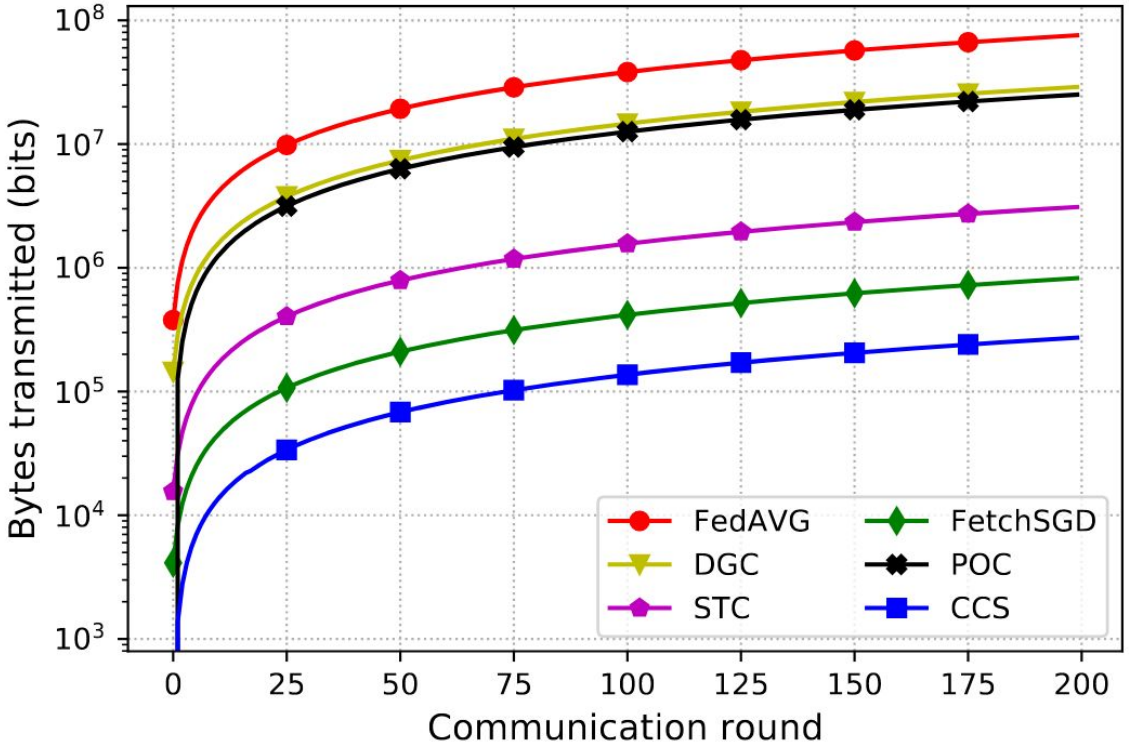
CCS – Compressed Client Selection

13: **LocalTrain(n, w):** ▷ *When client* $n \in S \mid S \subseteq N$ *receives a train request*
14: **for** each epoch $\in 1, 2, \ldots$ **do**
15: $w_n \leftarrow f(w_t, d_n)$ ▷ *fit the model with local data*
16: **end for**
17: $g_n \leftarrow getGradients(w_n)$ ▷ *compute gradients*
18: $w_c^n \leftarrow CountSketch(g_n)$ ▷ *apply the compression method*
19: $send2Server(w_c^n)$ ▷ *send the compressed model to the central server*

CCS – Compressed Client Selection



CCS – Compressed Client Selection



Future Work

- Improve Fairness
- Improve robustness
- Downstream compression
- Verify Secure Aggregation
- Client Selection