Sketch to Skip and Select: Communication Efficient Federated Learning using Locality Sensitive Hashing

Georgios Kollias , **Theodoros Salonidis** , **Shiqiang Wang**
IBM T. J. Watson Research Center
{gkollias, tsaloni, wangshiq}@us.ibm.com

Abstract

We introduce a novel approach for optimizing communication efficiency in Federated Learning (FL). The approach leverages sketching techniques in two complementary strategies that exploit similarities on the data transmitted during the FL training process to identify opportunities for skipping expensive communication of updated models in training iterations, and dynamically select subsets of clients hosting diverse models. Our extensive experimental investigation on different models, datasets and label distributions, shows that these strategies can massively reduce downlink and uplink communication volumes by factors order of $100\times$ or more with minor degradation or even increase of the accuracy of the trained model. Also, in contrast to baselines, these strategies can escape suboptimal descent paths and can yield smooth non-oscillatory accuracy profiles for non-IID data distributions.

1 Introduction

Communication efficiency is a crucial issue in Federated Learning (FL), especially when large deep learning models are employed. Existing techniques for reducing the computation and communication overhead of FL focus on selecting a small fraction of clients in each FL training round or transmitting compressed models instead of original models. However, most of these techniques are either based on a simple randomized procedure, such as random client selection [Bonawitz *et al.*, 2019], or approaches such as magnitude-based gradient compression [Konečný *et al.*, 2016]. These methods do not take data characteristics into account and ignore the fact that data at clients are often non-independent and/or non-identically distributed (non-IID). This causes issues such as low accuracy or bias as useful data on certain clients tends to be largely ignored by the FL training process. In addition, as we empirically show in this paper, random client selection can also cause instabilities in the convergence process for non-IID distributions.

Therefore, a largely open question is: *how to design communication efficient FL training procedures that are data-aware and can incorporate characteristics of non-IID data distribution?* We address this problem in this paper. The core idea is to compute sketches of useful information obtained during the FL process, and use these sketches to determine when to share the model parameter vector or the gradient between server and clients and which clients to select. The benefit of this approach is that the sketches capture important knowledge related to data distributions and progress of local model updates, which is not possible using existing techniques.

Our main contributions are as follows:

- We introduce a novel predicate that decides the proximity of models hosted at different nodes of an FL system, in a communication efficient way. The predicate is a communication skipping mechanism: if two remote models are approximately "close" do not share them. We implement this predicate based on Locality Sensitive Hashing (LSH) sketching techniques.
- We propose and implement a new method for client selection in FL that leverages sketching combined with clustering.
- We integrate our techniques in an FL simulation framework and evaluate their performance through extensive training experiments over six configurations of real datasets and data distributions and different parameters. We observe massive (order $100\times$) downlink and uplink communication savings over state of the art baselines without classification accuracy degradation. In some cases we obtain improvements in *both* accuracy and communication. For some non-IID data distributions we also demonstrate that in contrast to baselines, our client selection method avoids convergence issues such as fluctuations in accuracy during training.

2 Related Work

Communication efficiency is an extensively studied topic in general distributed learning from a theoretical point of view. It has also been addressed in applications such as distributed learning for data centers and more recently in the context of FL. In this section, we will focus on techniques that have been applied to FL context and are most related to this paper.

Local updating methods. Local updating methods aim to reduce the total number of communication rounds by doing more computations. The most commonly used method for FL is Federated Averaging (FedAvg) [McMahan *et al.*, 2017], a method based on averaging local stochastic gradient descent

(SGD) updates. FedAvg reduces communication overhead by doing multiple local SGD updates in parallel per communication round. FedProx improves upon FedAvg for the case of heterogeneous data [Li *et al.*, 2020]. The number of FedAvg SGD updates in these techniques is typically fixed and not adaptive to changing data. This can create divergence issues (if set too high) or high communication overhead (if set too low). The work of [Wang *et al.*, 2019] proposes adaptive techniques for deciding the number of local SGD updates and frequency of communication to server. However, it does not exploit data similarities between clients and server and across clients.

FL training using model compression. There exist several approaches which perform FL training using compressed model parameters or gradients in order to reduce communication overhead, at the potential expense of accuracy. These schemes use different compression techniques such as forcing the updating models to be sparse and low-rank or performing quantization with structured random rotations [Konečný *et al.*, 2016]. In [Li *et al.*, 2019], Li et. al. proposed `DiffSketch` which compresses transmitted messages via sketches to achieve communication efficiency and privacy benefits. For independent and identically distributed (IID) *MNIST* data with a large number of clients, they show high communication benefits, but these come at the expense of accuracy. Most approaches reduce overhead on the uplink direction from clients to server. The work in [Caldas *et al.*, 2018] uses lossy compression and dropout to reduce server-to-device downlink communication. In `FetchSGD` [Rothchild *et al.*, 2020] the clients sketch their gradients using Count Sketch [Charikar *et al.*, 2002],which is a randomized data structure, before transmitting to the server. In [Chen *et al.*, 2021] they similarly leverage count sketches for compression but under distributed differential privacy (DP) via secure aggregation (SecAgg).

Our approach significantly differs from the above approaches. It does not perform FL training based on compressed model parameters. Instead, it uses a special type of sketches based on LSH to decide on client selection and transmission of non-compressed model parameters. Thus, the goal is to minimize communications while using the necessary non-compressed data to maintain high accuracy by exploiting data similarities during the training process. In addition the above approaches typically demonstrate communication reduction but may yield significant accuracy loss. In contrast, our approach can yield massive communication overhead reduction (up to $100\times$) for non-IID distributions with minimal or zero accuracy loss. Our approach is also based on a generic sender/receiver predicate and is directly applicable to both uplink and downlink directions. It is also orthogonal to the existing compression-based training schemes and can be used in conjunction with them for further communication gains, if desirable.

Client selection methods. Client selection methods seek to reduce the number of participating clients. FedAvg uses random selection [Bonawitz *et al.*, 2019] which is used predominantly in FL settings in practice. There exist also works which select subset of clients based on device resource requirements [Wang *et al.*, 2020; Jin *et al.*, 2020; Nishio and Yonetani, 2019]. These techniques do not consider data characteristics. The investigation in [Fraboni *et al.*, 2021] proposes a data-aware client selection approach that is similar to ours at a high level. This approach involves hierarchical clustering of the clients based on a similarity metric on the gradients and sampling from these clusters. However, it incurs massive communication overhead during the clustering step because it requires all clients to send to the server their gradients (which have the same high dimensionality of model parameters). In contrast, in our `sketch-to-select` approach we use sketches of the model parameters which incur very low communication overhead in the clustering step. In addition, our `sketch-to-skip` strategy reduces communication overhead on both uplink and downlink by skipping client model updates to server and server global model transmission to the clients, respectively.

3 Methods

We leverage sketching techniques for informing our decisions of *when* to communicate model parameters and *which* clients to compute with. In particular, we vectorize the learning parameter tensors of a local model snapshot during training and compute a locality sensitive hash of the concatenation of vectors. For a locality sensitive hashing function $h(\cdot)$ and starting vectors $\vec{a}$ and $\vec{b}$, their hashes $h(\vec{a})$ and $h(\vec{b})$ are also vectors of reduced size, with the interesting property that they are close with high probability if $\vec{a}$ and $\vec{b}$ are close. This property motivates us to consider using such "short" hashes instead of the full model parameter tensors, when we need to *decide* whether two models, hosted at distributed agents, are approximately close (i.e. "similar") or not: agents compute, exchange and compare their "short" model hashes and then decide whether to communicate and share the actual models. This is a *communication efficient, model proximity predicate* and we use its Boolean output in two types of strategies:

1. **Sketch-to-Skip** skip sending the model snapshot, at execution points FL would normally send, when the predicate indicates that the model snapshot at the receiver side is "close".
2. **Sketch-to-Select** select a subset of computing clients that host models that are not "close" to each other.

3.1 Sketch-based communication skipping: `Sketch-to-Skip`

Our model proximity protocol uses a generic sender/receiver predicate and unlike most existing FL communication efficiency techniques, it can be applied to both uplink (clients to server) and downlink (server to clients) directions. Also it is a one-comparison protocol: projects and compares the projections. It is also modular in the sense that alternative sketching methods can be plugged-in: the work by [Datar *et al.*, 2004] is particularly relevant as they also minimize the distortion in l_2 norm.

Our model proximity predicate referred to as `lsh` in the sequel, is based on randomized projections (sketching) of *flattened* (i.e. vectorized and concatenated) model parameter tensors. Let $\vec{a}$ and $\vec{b}$, with d elements each, be the flattened tensors of the model paremeters at the perspective sender and receiver sides, which we wish to compare. In lsh, the sender generates k random vectors $\vec{r_1}, \vec{r_2}, \ldots, \vec{r_k}$, uniformly sampled from $(-1, 1)$ for given seed s and with d elements each. The vectors are organized in a $k \times d$ projection matrix that multiplies (projects) $\vec{a}$ to a k element vector $\vec{h_a}$. The sender sends the seed s and the sketching dimension k to the receiver so that it can generate an identical projection matrix, similarly compute a projection $\vec{h_b}$ of its $\vec{b}$ and send $\vec{h_b}$ to the sender. Finally the sender computes the relative norm difference of the two sketched vectors $\|\vec{h_a} - \vec{h_b}\|_2 / \|\vec{h_b}\|_2$: if this is smaller than the threshold parameter of this protocol, then the sender skips sending its $\vec{a}$.

Communication savings. The potential for communications savings is very high. When the proximity protocol decides that communication should be skipped (based on the lsh threshold parameter) only the hash of size k will have been sent instead of the full model of size d. As an example, the models in our experiments have hundreds thousands of parameters ($d = 238,510$ parameters for the FCNN model and $d = 555,178$ for the CNN model). The proximity protocol uses a vector of dimension $k = 100$. This can potentially yield dramatic communication savings of the order of $\frac{k}{d}$, i.e. 0.04% for FCNN and 0.02% for CNN when communication is skipped. We will quantify this potential in the experiments section.

Computation complexity. The computation of the hashes involves two steps. At the beginning of training iterations, at each client, a projection matrix is constructed which requires $O(dk)$ flops; this cost is amortized over all iterations. Then, for each iteration, this matrix is used to project the local model vector of size d at each agent, which also requires $O(dk)$ flops, since this is dense matrix vector multiplication (matvec). The work by [Ailon and Chazelle, 2006; Clarkson and Woodruff, 2017] can further reduce the complexity in generating the projection matrix or projecting the vectors, which according to [Konečný *et al.*, 2016] can be considered negligible compared to the computational complexity of local SGD training iterations within FL.

3.2 Sketch-based client selection: Sketch-to-Select

We introduce a "data-aware" scheme for client selection, where "data" corresponds to *sketches* of flattened model parameter tensors across all clients. As the last stage in iterations marked for updating the subset of selected clients (including c out of all n clients available), all n clients sketch their flattened model parameters and upload their resulting sketched vectors to the server. Then the server clusters the n sketched vectors into c clusters and randomly samples one vector per cluster. Clients for which their sketched vector was sampled are added to the new subset of selected clients.

The intention behind this scheme is the inclusion of clients hosting as diverse model parameters as possible. Parameters which are "close" will have their sketches being "close" and thus most probably they will land in the same cluster. This also means that selecting one representative model from each of c clusters - for a budget of c selected clients total - maximizes coverage of the distributed model parameter space distribution.

For sketching, we utilize the same type of randomized projection matrices as in lsh. We use ubiquitous Lloyd's algorithm [Lloyd, 1982] for clustering our n vectors (of k elements each, k the sketching dimension) into c clusters; also k-means++ [Arthur and Vassilvitskii, 2007] for initializing the coordinates of its c centroids. Interestingly, k-means++ has a analogous objective to what we are trying to do: build a set of centroids that are as far as possible from each other. So it becomes quite natural to use it for cluster initialization.

For the client slection iterations, the computation complexity for projecting is $O(dk)$ flops per client; at server side, Lloyd's algorithm incurs an overhead of $O(ncr)$ flops, where r is the number of rounds for centroid updates. Also, the communication complexity for uplink communication of the sketches is $O(nk)$.

3.3 Sketch to Skip and Select FL algorithm

The FedAvg FL training algorithm [McMahan *et al.*, 2017] consists of multiple rounds, where each round consists of the following steps: Running SGD model updates locally at the clients for E iterations. Then randomly selecting a subset of the clients to upload their updated model parameters to the server. The server aggregates the parameters of the clients by taking their mean and sends the updated global model to all clients.

Our *Sketch to Skip and Select FL* training algorithm is shown in Algorithm 1. At a high level it is based on two ideas for modifying FedAvg as follows.

The first idea is to perform model updates (uploading local client models/aggregation/downloading global server model) only if local client models are sufficiently different than the global server model. This is achieved using the sketch-to-skip strategy between server and clients, performing skip if the model of *all* selected clients is close to the global model at the server. When model updates are skipped, no client selection takes place. This approach can be seen in Algorithm 2.

The second idea is to use sketch-to-select data-aware scheme instead of FedAvg random client selection: Once the selected clients are able to upload their models, a new client selection can occur using our data-aware sketch-to-select algorithm. This is performed by all clients sending their model sketches to the server, the server clustering clients based on the sketches, and then selecting randomly a client from each cluster. The sketch-to-select strategy takes into account the clients' data heterogeneity issue as opposed to the FedAvg random client selection strategy which is oblivious to it. This approach can be seen in Algorithm 3.

More specifically, Algorithm 1 executes in multiple rounds $0, \ldots, T-1$. Each round starts with the sketch-to-skip Algorithm 2: the server sends the sketch of its global model to the clients; the clients update their model parameters by performing E local SGD steps, compare the updated model sketches with the global model sketch and send to server the outcome of the comparison; if *all* client models have not deviated from the global server model by more than δ, then the server decides to skip the uplink update, aggregation and downlink update steps (lines 13, 17, 21 in Algorithm 1, respectively) and client selection step (line 19 in Algorithm 1).

Client selection (`sketch-to-select` Algorithm 3) occurs at each $j * u$-th round, where u is a parameter and $j = 0, 1, \ldots$, provided that skip does not occur on that round.

Algorithm 1 Sketch to Skip and Select FL

```
 1: Input: number of all clients $N$, number of clusters $C$,
    number of rounds $T$, number of local updates $E$, learning
    rate $\eta$, skip step threshold $\delta$, sketching dimension $k$, set of
    client selection update steps $\mathcal{U}_T = \{0, u, 2u, \ldots, \lfloor \frac{T}{u} \rfloor u\}$
 2: At server:
 3: Initialize model parameters $\bar{\mathbf{w}}^0$ and set of selected clients
    $\mathcal{S}_0 = [N]$
 4: Transmit $\bar{\mathbf{w}}^0$ to all clients $i \in \mathcal{S}_0$
 5: for $t = 0, \ldots, T-1$ do
 6:   Execute Sketch-to-Skip strategy (Algorithm 2)
 7:   At clients:
 8:   for each client $i \in \mathcal{S}_t$ in parallel do
 9:     Comment: skip received as in Algorithm 2
10:     if value of skip received from server is True then
11:       Skip this round $t$: continue to round $t+1$
12:     else
13:       Transmit local model $\mathbf{w}_i^t$ to the server
14:     end if
15:   end for
16:   Server update:
17:   $\bar{\mathbf{w}}^{t+1} \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \mathbf{w}_i^t$
18:   if $t \in \mathcal{U}_T$ then
19:     Execute Sketch-to-Select strategy (Algo-
        rithm 3)
20:   end if
21:   Server broadcast $\bar{\mathbf{w}}^{t+1}$ to all clients
22: end for
```

4 Experiments

4.1 Experimental setup

Datasets and Data Distributions at Different Nodes. We experiment with three datasets: the original MNIST (referred to as *MNIST-O*) [LeCun *et al.*, 1998], Fashion-MNIST (referred to as *MNIST-F*) [Xiao *et al.*, 2017] and CIFAR-10 [Krizhevsky, 2009]. We consider two different ways of distributing the data into different nodes for each of the datasets. In Case 1, each data sample is randomly assigned to a node, thus each node has uniform (but not full) information. In Case 2, all the data samples in each node have the same label[1]. This represents the case where each node has non-uniform (non-IID) information, because the entire dataset has samples with multiple different labels.

Models. For *MNIST-O* and *MNIST-F* we train deep fully connected neural network (FCNNs) and for *CIFAR-10* deep convolutional neural networks (CNNs).[2] We use stochastic gradient descent (SGD) for training FCNNs and CNNs. The loss function is cross-entropy on cascaded linear and non-linear transforms [Goodfellow *et al.*, 2016].

Algorithm 2 `Sketch-to-Skip` strategy

```
 1: At server:
 2: $\bar{\mathbf{h}}^t \leftarrow$ Flatten and sketch global model $\bar{\mathbf{w}}^t$; $\bar{\mathbf{h}}^t \in \mathbb{R}^k$
 3: Transmit $\bar{\mathbf{h}}^t$ to all clients $i \in \mathcal{S}_t$
 4: At clients:
 5: for each client $i \in \mathcal{S}_t$ in parallel do
 6:   $\mathbf{w}_{i,t}^0 \leftarrow \bar{\mathbf{w}}^t$
 7:   for $j = 0, \ldots, E-1$ do
 8:     $g_i(\mathbf{w}_{i,t}^j) \leftarrow \nabla f_i(\mathbf{w}_{i,t}^j)$
 9:     $\mathbf{w}_{i,t}^{j+1} \leftarrow \mathbf{w}_{i,t}^j - \eta g_i(\mathbf{w}_{i,t}^j)$
10:   end for
11:   $\mathbf{w}_i^t \leftarrow \mathbf{w}_{i,t}^E$
12:   $\bar{\mathbf{h}}_i^t \leftarrow$ Flatten and sketch local model $\mathbf{w}_i^t$; $\bar{\mathbf{h}}_i^t \in \mathbb{R}^k$
13:   skip$_i$ ← False
14:   if $\|\bar{\mathbf{h}}_i^t - \bar{\mathbf{h}}^t\|_2 / \|\bar{\mathbf{h}}^t\|_2 < \delta$ then
15:     skip$_i$ ← True
16:   end if
17:   Transmit skip$_i$ to server
18: end for
19: At Server:
20: skip ← False
21: if skip$_i$ is True for all $i \in \mathcal{S}_t$ then
22:   skip ← True
23: end if
24: Transmit skip to all clients $i \in \mathcal{S}_t$
```

Training and Control Parameters. For local training at each client, we use a static learning rate of $\eta = 0.05$ for *MNIST-O* and *MNIST-F* and $\eta = 0.01$ for *CIFAR-10*. The mini-batch size is $B = 100$. We train for $T = 1000$ rounds. We simulate a client-server FL system of $N = 50$ clients using synchronous $E = 1$ steps of distributed gradient descent [Chen *et al.*, 2016] and select $C = 10$ (out of the 50) clients for the client selection runs. Selected clients are then elected every $u = 100$ iterations (frequency of client selection) for all applicable cases. For the sketching dimension k, we use the values of 100 and 10 for sketch-based communication skipping and client selection, respectively.

Comparison Baseline. We compare our methods against FedAvg [McMahan *et al.*, 2017] with the same training and control parameters (η,E,B,T,N,C) as above.

Metrics. We collect downlink and uplink communication volumes during training and the accuracy value at its end, for all attempted combinations and setup parameters (proximity primitives, thresholds and client selection intervals when in client selection mode). Then we compute the *percent relative error* $100 * (x - y)/y\%$ for the accuracy (where x is the classification accuracy of our method and y of the comparison

[1]When there are more labels than nodes, each node may have data with more than one label, but the number of labels at each node is no more than the total number of labels divided by the total number of nodes rounded tothe next integer

[2]The FCNN has 3 layers with the following structure: 784×300 Fully Connected → 300×10 Fully Connected → Softmax. The CNN has 9 layers with the following structure [Tensorflow, 2020]: $5 \times 5 \times 32$ Convolutional → 2×2 MaxPool → Local Response Normalization → $5 \times 5 \times 32$ Convolutional → Local Response Normalization → 2×2 MaxPool → 2048×256 Fully connected → 256×10 Fully connected → Softmax.

Algorithm 3 `Sketch-to-Select` strategy

```
1: At clients:
2: $\bar{\mathbf{h}}_i^t \leftarrow$ All clients $i \in [N] \setminus \mathcal{S}_t$ flatten and sketch their local models $\mathbf{w}_{i,t}$
3: All clients $i \in [N]$ transmit their local model sketches $\bar{\mathbf{h}}_i^t$ to the server
4: At server:
5: Server computes clusters $\mathcal{A}^t := \{\mathcal{A}_1^t, \ldots, \mathcal{A}_C^t\}$ from $\bar{\mathbf{h}}_i^t$, $i \in [N]$ using Lloyd's algorithm with k-means++ initialization
6: Server samples $\mathcal{S}_t \in [N]$ by drawing one client from each cluster $\mathcal{A}_i^t, i \in [C]$ uniformly at random
7: Server broadcasts $\bar{\mathbf{w}}^{t+1}$ to sampled clients
```

baseline), also referred as *accuracy increase* and the *percent ratio* $100 * x/y$ for the downlink and uplink communication volumes (where x refers to our method and y to the comparison baseline), also referred to as *overhead ratio*[3].

4.2 Results

Figure 1 depicts the communication savings (overhead ratio) vs. accuracy increase obtained with our proximity primitive and client selection strategies, with respect to the `FedAvg` with random client selection baseline, for various parameter settings of our strategies on the 6 configurations. We observe that for each configuration there exist several parameter settings that yield very low uplink/downlink overhead ratios (10% or much lower) and small negative (-5%) or positive accuracy increase.

Figure 2 shows boxplot statistics for accuracy increase, downlink and uplink overhead ratio for each configuration in Figure 1. We observe that the median overhead ratio across the six configurations ranges between $[0.21\%, 28.28\%]$ ($[476x, 3.5x]$ savings) for downlink and $[0.57\%, 31.25\%]$ ($[175x, 3.2x]$ savings) for uplink communication. Median accuracy increase ranges between $[-6.47\%, 34.22\%]$ across the 6 configurations. For example, in CIFAR-10 Case 2 configuration, the median overhead ratio (across all `lsh` parameter settings we used) for downlink is 0.21% (476x savings) and for uplink 0.57% (175x savings) while median accuracy increase is 5.6%.

Figure 3 compares the accuracy profiles of our approach for a selected parameter set and FedAvg with random client selection baseline for each of the 6 configurations. In 5/6 configurations our approach yields higher accuracy and in CIFAR-10, Case 1 it is tracking closely the baseline accuracy. A few more interesting observations are in place.

Our approach exhibits a step-wise accuracy evolution. This is due to skipping sharing model parameters during some iterations until the models become different enough to resume sharing. Essentially the number of steps equals the number of total iterations minus the number of iterations where communication was skipped.

[3]A positive (negative) *accuracy increase* means that our method attains better (worse) accuracy than the baseline, by the indicated percent, so *the more positive the better*. Similarly 1%, 5%, 10%, 50% *overhead ratio* values correspond to $100\times$, $20\times$, $10\times$, $2\times$ communication savings compared to baseline, respectively: *the smaller the better*, with the baseline at 100%.

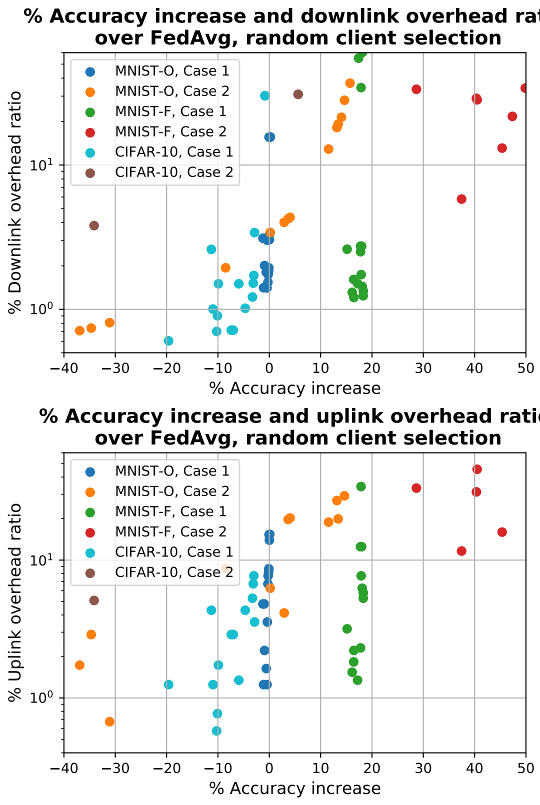

Figure 1: Overhead ratio for downlink (top panel) and uplink (bottom panel) communication (vertical axis) vs accuracy increase (horizontal axis). Baseline is `FedAvg` with random client selection.

Random client selection accuracy profiles are "oscillatory" for Case 2 distribution (Figure 3, bottom row.) Random client selection will select clients without taking into account the latent affinity in the local models for some of the labels. In contrast, our client selection will group affine clients and select a single representative from the group, thus ensuring that affine clients will not be selected for the same selection interval (in the limit: avoid repetition of samples) and save their spot for inclusion of models that are not close (in the limit: include new samples). Therefore, our sketch-based client selection results in stable accuracy profiles for these cases and this reflects the crucial role of clustering as a mechanism to diversify.

***MNIST-F* gets significant accuracy boost in addition to communication savings (middle panes in Figure 3.)** Effectively, sketching seems to be providing a mechanism to escape suboptimal descent paths during training for some datasets.

5 Conclusions

Sketching has been used in FL to compress model representations prior to sending, with the intention to directly mix the compressed representation into the computation of the model update at the receiving side [Konečnỳ *et al.*, 2016]. In our work, different sketches based on locality sensitive hashing

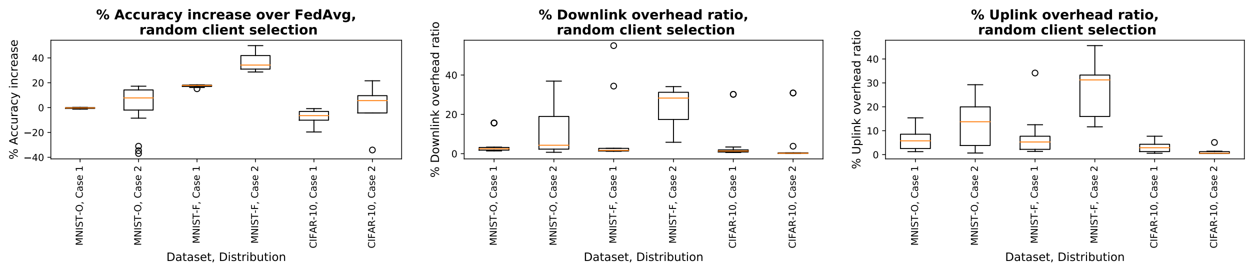

Figure 2: Box plots for accuracy increase (left panel) and overhead ratio for downlink (middle panel) and uplink (right panel) communication for various dataset and distribution combinations as in Figure 1. Baseline is `FedAvg` with random client selection.

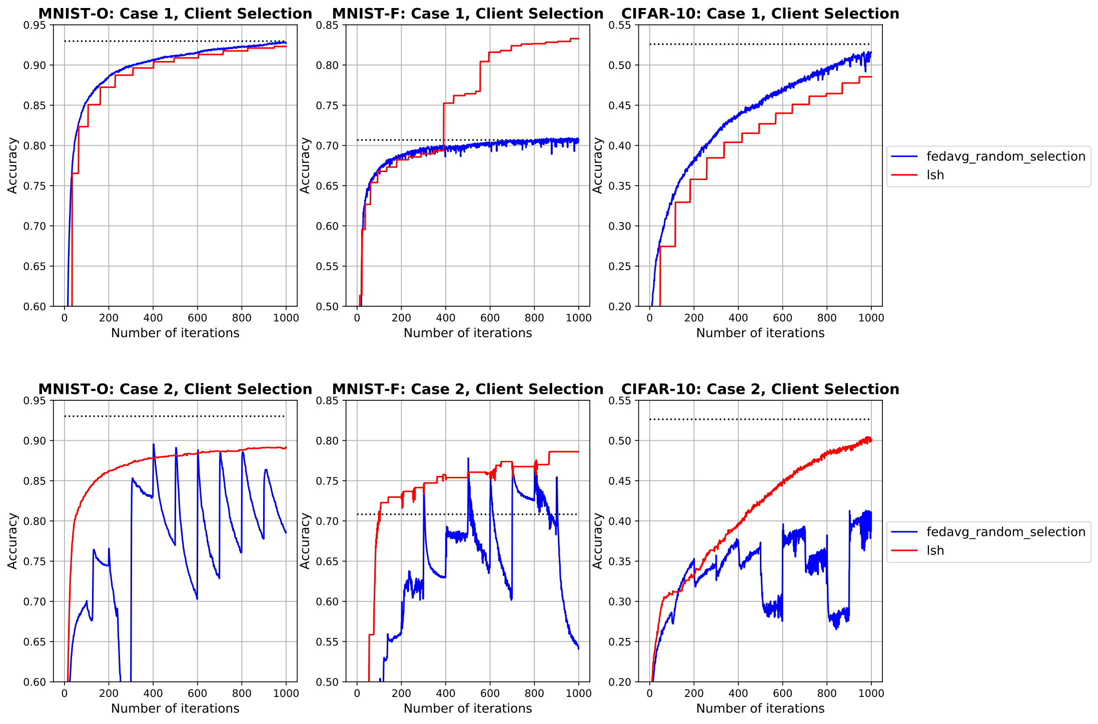

Figure 3: Classification accuracy with client selection: Case 1 (top row); Case 2 (bottom row).

are used in completely orthogonal and indirect ways: to decide whether communicating the model should occur in the first place and which clients to engage in the computation. We have empirically demonstrated that this approach can massively reduce downlink and uplink communication volumes by factors order of $100\times$ or more with minor degradation or even increase of the accuracy of the trained model. We also empirically identified cases where, in contrast to the baseline, our strategies provide a mechanism to escape suboptimal descent paths and can yield smooth accuracy profiles for non-IID data distributions.

References

[Ailon and Chazelle, 2006] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, 2006.

[Arthur and Vassilvitskii, 2007] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[Bonawitz *et al.*, 2019] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. In *Systems and Machine Learning (SysML) Conference*, 2019.

[Caldas *et al.*, 2018] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[Charikar *et al.*, 2002] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[Chen *et al.*, 2016] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*, 2016.

[Chen *et al.*, 2021] Wei-Ning Chen, Christopher A Choquette-Choo, and Peter Kairouz. Communication efficient federated learning with secure aggregation and differential privacy. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

[Clarkson and Woodruff, 2017] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[Fraboni *et al.*, 2021] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *Proc. International Conference on Machine Learning*, 18–24 Jul 2021.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[Jin *et al.*, 2020] Yibo Jin, Lei Jiao, Zhuzhong Qian, Sheng Zhang, Sanglu Lu, and Xiaoliang Wang. Resource-efficient and convergence-preserving online participant selection in federated learning. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[Konečnỳ *et al.*, 2016] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li *et al.*, 2019] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972*, 2019.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Machine Learning and Systems (MLSys) Conference*, 2020.

[Lloyd, 1982] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[Nishio and Yonetani, 2019] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE International Conference on Communications (ICC)*, pages 1–7, 2019.

[Rothchild *et al.*, 2020] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[Tensorflow, 2020] Tensorflow. Convolutional neural network (cnn). https://www.tensorflow.org/tutorials/images/cnn, 2020. Online.

[Wang *et al.*, 2019] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[Wang *et al.*, 2020] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1698–1707, 2020.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.