QSFL: A Two-Level Uplink Communication Optimization Framework for Federated Learning

Liping Yi [1] **Gang Wang** [1] **Xiaoguang Liu** [1]

Abstract

In cross-device Federated Learning (FL), the communication cost of transmitting full-precision models between edge devices and a central server is a significant bottleneck, due to expensive, unreliable, and low-bandwidth wireless connections. As a solution, we propose a novel FL framework named *QSFL*, towards *optimizing FL uplink (client-to-server) communication at both client and model levels*. At the client level, we design a *Qualification Judgment (QJ)* algorithm to sample high-qualification clients to upload models. At the model level, we explore a *Sparse Cyclic Sliding Segment (SCSS)* algorithm to further compress transmitted models. We prove that QSFL can converge over wall-to-wall time, and develop an optimal hyperparameter searching algorithm based on theoretical analysis to enable QSFL to make the best trade-off between model accuracy and communication cost. Experimental results show that QSFL achieves state-of-the-art compression ratios with marginal model accuracy degradation.

1. Introduction

In traditional FL, participating clients upload local full-precision models to the server for aggregating without local data sharing. When thousands of clients collaboratively train one GB-level model, uplink transmission can reach TB-level (Sattler et al., 2019b). However, expensive and unreliable wireless connections between edge devices and the server often have a lower rate than the Ethernet connection in a data center (Yao et al., 2019; Zhang et al., 2020). And large-scale deep networks often have lots of redundant parameters (Kairouz et al., 2019; Chen et al., 2018b; Yi et al., 2020), transmitting all local full-precision models leads to communication overhead being a major bottleneck. Besides, the uploading and downloading speed of internet connection are highly asymmetrical: the former is slower at least five times than the latter (Konečný et al., 2016), so compressing client-to-server transmission traffic yields greater benefits.

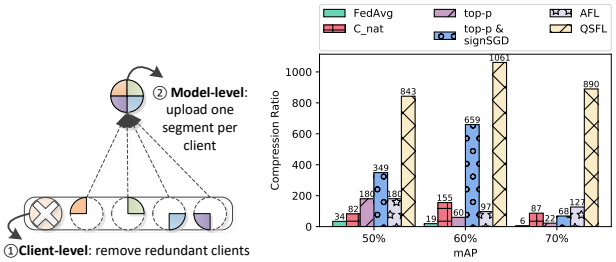


Figure 1: Left: insights of QSFL. Right: QSFL outperforms alternatives in the compression ratio under 50%, 60% and 70% target mAP on the FEMINIST dataset, respectively.

Existing communication-effective FL schemes mainly involve: delaying communication, sampling clients, encoding models, sparsification, quantization. Since these methods either have no theoretical convergence guarantee or have limited compression ratios, more advanced schemes should be designed to balance communication cost and convergence.

In this work, we propose a novel communication-efficient framework named QSFL, which optimizes FL's uplink communication cost by: a) first sampling high-qualification clients to upload model updates (QJ algorithm), and b) further compressing each uploaded model to one segment in each round (SCSS algorithm). We prove its convergence and explore how to search its optimal hyperparameters based on theoretical analysis. As shown in Fig. 1, QSFL fulfills the state-of-the-art compression ratio than advanced communication-efficient methods when achieving the same target accuracy, promoting a better trade-off between communication cost and model accuracy.

Contributions. Our main contributions are as follows:

- We propose a novel FL framework named QSFL to reduce FL uplink communication overhead at both client-level and model-level.

[1]Nankai-Orange D.T. Joint Lab, College of Computer Science, Nankai University, Tianjin, China. Correspondence to: Gang Wang <wgzwp@nbjl.nankai.edu.cn>, Xiaoguang Liu <liuxg@nbjl.nankai.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

- We prove the convergence rate of QSFL and develop a hyperparameter searching algorithm to configure optimal hyperparameters for QSFL.
- Extensive experimental results verify the advance of QSFL in uplink communication overhead optimization.

2. Related Work

At present, existing FL communication optimization methods can be summarized into the following categories:

(A) Delaying Communication. FedAvg (McMahan et al., 2017) delays communication by increasing local iterations to accelerate the whole training process. FedPAQ (Reisizadeh et al., 2020) aggregates periodically local quantized model updates to reduce communication cost.

(B) Sparsification. Sparsification replaces full-precision models with appropriate sparse representations. Strom (2015); Tsuzuku et al. (2018) select parameters with magnitudes greater than a threshold to transmit, but it's hard to set an appropriate threshold. Gradient dropping (Aji et al., 2017) samples parameters with top-$p\%$ magnitudes. Wangni et al. (2018) introduces one convexity formula to ensure that compressed gradients are unbiased estimates of true gradients. The variance-based sparsification in Tsuzuku et al. (2018) only uploads gradients with magnitudes greater than variances. Slim-DP (Sun et al., 2018) samples parameters with top-$p\%$ magnitudes and randomly selects $\epsilon\%$ from remaining parameters. HeteroFL (Diao et al., 2021) allows each client to upload its local model's sub-network adaptive to system resources to the server, implementing the compatibility with heterogeneous sub-networks and improving uplink communication's efficiency. Although top-$p\%$ sparsification is robust to non-iid data (Sattler et al., 2019b), model accuracy drops obviously as the compression ratio rises, and it also lacks theoretical guarantees for convergence.

(C) Quantization. Quantization represents model parameters with fewer bits. SignSGD (Bernstein et al., 2018) quantizes gradients into binary symbols, it performs poorly with non-iid data (Sattler et al., 2019b). FTTQ (Xu et al., 2020) quantizes 32-bit model parameters into 2-bit ternary symbols. Compared with binary and ternary quantization, multi-bit quantization (Cai et al., 2019; Guo et al., 2017; Leng et al., 2018; Lin et al., 2017; Zhou et al., 2016) has lower quantization errors. Caldas et al. (2018a) introduces lossy compression with quantization and FL-dropout to reduce communication cost. C_{nat} (Horvath et al., 2019) quantizes each parameter t to $\mathrm{sgn}(t) * 2^n$ and only uploads the symbol $\mathrm{sgn}(t)$ and exponent power n. TernGrad (Wen et al., 2017), QSGD (Alistarh et al., 2017) and ATOMO (Wang et al., 2018) quantize parameters in an unbiased manner. Although quantization has theoretical convergence guarantee, its upper bound of compression ratio is only $32\times$.

(D) Combinations of Sparsification and Quantization. STC (Sattler et al., 2019b) proposes a combination of top-$p\%$ sparsification, ternary quantization, and Golomb coding to realize two-way compression. SBC (Sattler et al., 2019a) combines communication delay, sparsification, and binary quantization to compress two-way transmitted models.

(E) Parameters Encoding. SKETCHED-SGD (Ivkin et al., 2019) compresses local gradients into sketches, but secondary communication yields extra cost. FetchSGD (Rothchild et al., 2020) introduces momentum and error accumulation on the server to alleviate the impacts of sketched representation. DiffSketch (Li et al., 2019) takes a better trade-off among accuracy, privacy, and communication cost. Alistarh et al. (2017) encodes quantized gradients with Elias coding. Weightless (Reagen et al., 2018) combines clustering quantization and compact coding of bloom filters to achieve model compression. PowerSGD (Vogels et al., 2019) decomposes models into smaller matrices through low rank while using a hierarchical "all-reduce" aggregation. Chen et al. (2021) proposes a combination of sampling client, random lattice quantization, and wireless resource allocation to optimize FL communication.

(F) Client Sampling. AFL (Goetz et al., 2019) actively samples clients with large model loss to speed up convergence. CMFL (Wang et al., 2019) calculates each client's relevance to the global model and forbids clients with relevance lower than a threshold to upload parameters. LAG (Chen et al., 2018a) skips selecting redundant clients whose gradients during two consecutive rounds have little change. LAQ (Sun et al., 2019) quantizes the gradients before applying LAG.

Our Insights. Inspired by *client sampling* and *sparsification*, we propose QSFL to sample clients with high qualifications judged by designing a more comprehensive judgment (QJ) rule, and further compress uploaded models through designing a sparse cyclic sliding segment (SCSS) algorithm. Limited to client-to-server bandwidths in parameter-server (PS) FL, Hu et al. (2019) proposes a decentralized (P2P) FL framework, which designs a *segmented gossip* approach to *maximize* bandwidth utilization between workers (clients). It allows each worker to *randomly* pull *replica* segments from *multiple* other workers. Whereas, our SCSS aims to *minimize* communication cost by allowing each client to upload *only one well-specified* segment based on the sliding rule in PS FL. Hence, the development scenarios, optimization goals and transmission protocols of them are all distinct.

3. Methods

In this section, we first formulate the communication-efficient FL problem, then present our QSFL framework, finally prove the convergence of QSFL and propose a hyperparameter searching algorithm based on theoretical analysis.

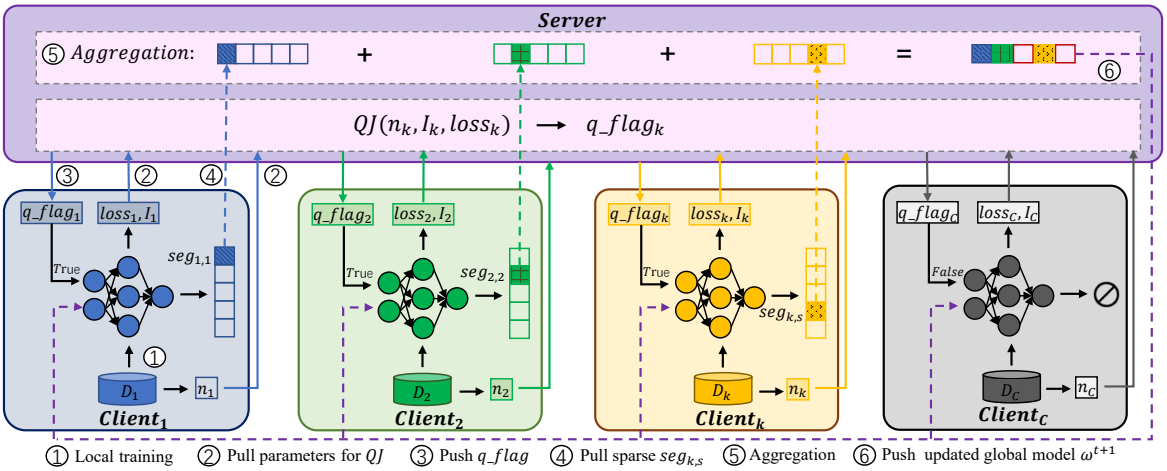


Figure 2: A complete workflow of QSFL framework in one round. ①: clients train local models on local datasets, ②: clients compute and upload the three key parameters for qualification judgment, ③: the server calculates and sends qualification flags back to clients, ④: clients with uploading qualification (flag is set to **true**) upload one particular sparse segment to the server, ⑤: the server aggregates segments with same id, ⑥: the server broadcasts the updated global model to all clients.

3.1. Problem Formulation

Background with FedAvg (McMahan et al., 2017), we assume that C clients join in FL and the objective of traditional FL is to minimize the mean loss of local models:

$$\min_{\omega \in \mathbb{R}^d} F(\omega) = \sum_{k=1}^{C} \frac{n_k}{n} f^k(\omega_k), \tag{1}$$

where n_k is the k-th client's data volume, n is the total data volume held by all clients, ω is the global model, ω_k is the local model of the k-th client, $f^k(\omega_k)$ is the loss function of ω_k and can be computed by:

$$f^k(\omega_k) = \frac{1}{n_k} \sum_{i \in D_k} f_i^k(\omega_k), \tag{2}$$

where D_k is the local dataset of the k-th client, $f_i^k(\omega_k)$ is the loss of i-th data instance on ω_k, i.e.,

$$f_i^k(\omega_k) = \ell(x_i^k, y_i^k; \omega_k). \tag{3}$$

With the above formulation, we approach a communication-efficient FL by: **a) sampling partial clients to upload models (reducing C)**, and **b) compressing uploaded ω_k**.

3.2. QSFL Framework

This section describes how to optimize client-to-server communication in our two-level QSFL framework. In each round, it first uses a QJ algorithm based on contribution and relevance to select high-qualification clients to upload model updates. Second, each qualified client uploads one particular sparse segment controlled by the SCSS algorithm to the server. Fig. 2 displays the complete training and communication pipeline of QSFL in one round.

3.2.1. Qualification Judgment (QJ)

We assume that the server and clients joining in FL are both honest and trustworthy. Clients with different numbers and distributions of instances make different contributions to the global model. We regard clients with low contribution as "redundant clients" and forbid them from uploading model updates. Dropping models of redundant clients has marginal effects on the accuracy of the aggregated global model but effectively saves uplink communication bandwidth at the *client level*. Next, we detail how to judge redundant clients.

Definition. We define one client's uploading qualification from accelerating FL training and ensuring the convergence of the global model as follows:

$$\begin{aligned} q_k &= \beta \cdot contribution_k + (1-\beta) \cdot relevance_k \\ &= \beta \cdot \frac{loss_k / n_k}{\sum_{i=1}^{C} loss_i / n_i} \\ &+ (1-\beta) \cdot \frac{\sum_{j=1}^{|\omega_k^l|} \mathbb{I}(sgn(\omega_{k,j}^l) == sgn(\omega_j^g))}{|\omega_k^l|}, \end{aligned} \tag{4}$$

where $\mathbb{I}(\cdot)$ is an indicator function, $sgn(\cdot)$ is a sign function, $\omega_{k,j}^l$ is the j-th parameter of the k-th client's *local* model, ω_j^g is the j-th parameter of the *global* model. Since $contribution_k$ and $relevance_k$ belong to [0,1], $q_k \in$ [0,1].

The qualification judgment process is executed by the *server*. As shown in Fig. 2, all clients upload three key parameters $relevance_k$ (a.k.a I_k), n_k, $loss_k$ to the server after local training, then the server updates the qualification list $Q = [q_1, \ldots, q_k, \ldots, q_C]$ with received parameters and Eq. (4), and sorts items of Q in descending order. The last $(1-\alpha\%)$ of the clients are judged as redundant, and their qualification flags q_flag_k are set to **false**, otherwise **true**. Only clients with q_flag_k= **true** are allowed to upload model updates.

Computing Contribution. We measure each client's contribution through normalized $loss_k/n_k$. The larger mean loss makes the training process gain more by gradient descent optimizer (Goetz et al., 2019). FL training aims to reduce the mean loss of all clients, so choosing clients with larger loss can *speed up convergence*, while dropping several clients with low loss has marginal impacts on the loss of the aggregated global model.

Computing Relevance. We compute the relevance between the received global model and trained local models by (number of parameters with the same signs on the same coordinates) / (total number of model parameters). Wang et al. (2019) has proved that sign consistency of local models and the global model can guarantee consistent gradient directions, so the higher relevance indicates the local model's gradient direction tends more to the global model. Hence, sampling high-relevance clients can *ensure convergence*.

Communication Cost. Each client communicates three 32-bit floating points (I_k, n_k, $loss_k$) and one integer (q_flag_k) with the server, so it has negligible communication cost.

Computational Cost. For local computation, n_k is the k-th client's data volume, $loss_k$ is the byproduct of local training, both of the two requires no extra computation. Only I_k is computed by comparing the *signs* of parameters between the k-th client's local model and the global model. We take one *fully connected layer* of a LeNet model as an example, its input and output dimensions are x, y, then *one forward operation during training* consumes $64xy$[1] bits FLOPs. Whereas, computing the *relevance* of this layer between the k-th local model and the global model only consumes xy[2] bits FLOPs, since relevance is computed through comparing parameters' *signs* (1 bit) by ordinate. The computational cost of one forward operation on *convolutional layers* are even higher than computing relevance. Hence, the computational cost of I_k can be neglected and marginally affects on QJ's efficiency and practicality.

3.2.2. Sparse Cyclic Sliding Segment (SCSS)

Client Uploading. SCSS first flattens each local model of clients with uploading qualification to one-dimensional vector representation, and *evenly* divides it into NS segments with the *same* size, where $1 \leq NS \leq NC$ (the number of qualified clients), NS = 1 means the full model vector is transmitted. Then *each client uploads only one specific segment per round, and the uploaded segment's id is controlled by* $(client\_id + round)\%NS$*, which guarantees the uploaded* seg_id *always cyclically slides one segment forward relative to that in the last round.* To ensure that

[1]Here $(x+(x-1)+1)\cdot y = 2xy$ times calculations are required, and parameters are 32-bit floats, so consuming $2xy \cdot 32 = 64xy$ bits FLOPs.

[2]xy is the total number of parameters in this FC layer.

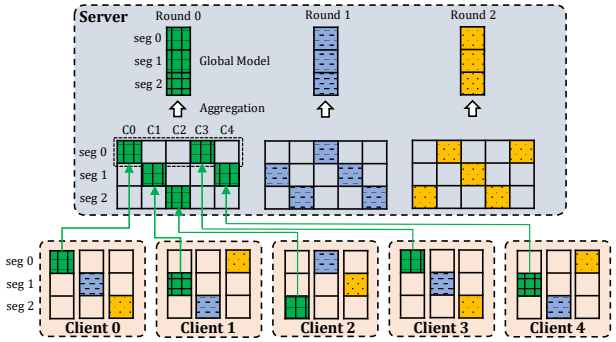


Figure 3: A toy example of SCSS.

segments with any id are uploaded by at least one client in a round, NS should be no more than NC, e.g., assuming that 8 clients upload models and 9 segments with different seg_id are required to be uploaded, one segment of all will not be uploaded by any client, leading to partial information missing. Due to only uploading one segment per client per round, SCSS requires more rounds to achieve the same convergence rate with FedAvg.

Sparse Trick. Each uploaded segment still involves redundant parameters, top-$p\%$ sparsification (Aji et al., 2017) can be exploited as a *trick* to further compress each segment.

Server Aggregating. The server first decodes received sparse segments, then averages segments with the same seg_id, finally splices NS averaged segments to reassemble the updated global model. The novel aggregation rule can be defined as follows:

$$\omega_g^t = [\frac{\sum\limits_{c\in c^0} seg_{c,0}^t}{|c^0|}, \frac{\sum\limits_{c\in c^1} seg_{c,1}^t}{|c^1|}, ..., \frac{\sum\limits_{c\in c^{NS-1}} seg_{c,NS-1}^t}{|c^{NS-1}|}], \quad (5)$$

where ω_g^t is the aggregated *global* model in the t-th round, and c^i is a *set* of clients who upload the i-th segment. The whole SCSS algorithm is described in Algo. 1.

Toy Example. For ease of understanding SCSS, we take 5 clients (NC = 5) and 3 segments (NS = 3) as an example shown in Fig. 3. $seg_{c,s}^t$ denotes the c-th client uploads its s-th segment in the t-th round. In the 0-th round, $client_0$ uploads the segment with seg_id = $(0+0)\%3$ = 0, i.e., $seg_{0,0}^0$. Similarly, $client_1$ uploads $seg_{1,1}^0$, $client_2$ uploads $seg_{2,2}^0$, $client_3$ uploads $seg_{3,0}^0$, and $client_4$ uploads $seg_{4,1}^0$. seg_0s are provided by $client_0$ and $client_3$, seg_1s are uploaded by $client_1$ and $client_4$, and seg_2 is only provided by $client_2$. The server then averages $seg_{0,0}^0$ and $seg_{3,0}^0$ as the updated seg_0, and so on. Finally, the server splices updated seg_0, seg_1 and seg_2 as the latest global model and sends it to all clients. In the next round, each client's uploaded segment cyclically slides forward by one segment, which ensures that each client can upload all of its segments after multiple rounds.

Algorithm 1 SCSS Algorithm

Client k does:
Receive the global model ω^t from the server
$\omega_k^t \leftarrow$ local training on ω^t
$s \leftarrow (k+t)\%NS$ // k: client id; t: round; s: segment id
$seg_{k,s} \leftarrow flatten(\omega_k^t)[s \cdot |\omega_k^t|/NS : (s+1) \cdot |\omega_k^t|/NS]$
masks, values of $sseg_{k,s} \leftarrow$ top-$p(seg_{k,s})$
Return masks, values of $sseg_{k,s}$

Server does:
Receive masks, values of $sseg_{k,s}$ from qualified clients
Initialize $SEG = [[sseg_0], \ldots, [sseg_{NS-1}]]$ to be empty
for $k = 0$ **to** $NC - 1$ **do**
  $sseg_{k,s} \leftarrow$ decode(masks, values of $sseg_{k,s}$)
  $SEG[s]$.append($sseg_{k,s}$)
end for
for $s = 0$ **to** $NS - 1$ **do**
  $SEG[s] = \text{avg}([sseg_{k_1,s}], [sseg_{k_2,s}], \cdots)$
end for
$\omega^{t+1} \leftarrow$ reshape(SEG, shape of original ω^t)
Return ω^{t+1}

3.3. Convergence Analysis and Proof

3.3.1. Intuitive Analysis

There are two points of SCSS worthy to note that: 1) Each client receives the complete global model from the server and then updates it locally, so all segments are always the *latest*. 2) The id of the segment uploaded by each client cyclically slides forward one as rounds rise, ensuring all parameters of one client's local model can be aggregated after multiple rounds. The above two points can ensure model convergence even if uploading sparse information.

Specifically, each client has uploaded NS segments with different id to the server after NS rounds, and the accumulated segments on the server during NS rounds are:

$$\begin{aligned}&\sum_{r=0}^{NS-1} \omega_g^r = \omega_g^0 + \omega_g^1 + ... + \omega_g^{NS-1} \\ &= [seg_{0,0}^0, seg_{1,1}^0, ..., seg_{NC-1,(NC-1)\%NS}^0] \\ &+ [seg_{0,1}^1, seg_{1,2}^1, ..., seg_{NC-1,NC\%NS}^1] + ... \\ &+ [seg_{0,NS-1}^{NS-1}, seg_{1,0}^{NS-1}, ..., seg_{NC-1,(NC+NS-2)\%NS}^{NS-1}].\end{aligned} \tag{6}$$

On the whole, the number of parameters server received during NS rounds in SCSS is the same as FedAvg in *one* round (e.g., in the 0-th round):

$$\begin{aligned}\omega_g^0 &= [seg_{0,0}^0, seg_{0,1}^0, ..., seg_{0,NS-1}^0] \\ &+ [seg_{1,0}^0, seg_{1,1}^0, ..., seg_{1,NS-1}^0] + ... \\ &+ [seg_{NC-1,0}^0, seg_{NC-1,1}^0, ..., seg_{NC-1,NS-1}^0].\end{aligned} \tag{7}$$

Each *column* in Eq. (6) corresponds to each *row* in Eq. (7), both of them represent the uplink communication traffic of the same client. In SCSS, the uploaded segments of each client during NS rounds cyclically slide forward, and corresponding gradients continuously decrease, which ensures convergence. In essence, SCSS has more local computation than FedAvg by delaying uploading remain segments.

3.3.2. Proof

The loss function of k-th client can also be represented by: $f^k(\omega_k) = \mathbb{E}_{B \subseteq D_k} f^k(\omega_k, B)$, B is a batch of dataset D_k and the batch size is $\mathbb{B}$. We first refer to the following assumptions made in Wang et al. (2020):

Assumption 3.1. Lipschitian gradient. The k-th client's local model gradient $\nabla f^k(\omega_k)$ is L-Lipschitian, i.e.,

$$\|\nabla f^k(\omega_k^{t+1}) - \nabla f^k(\omega_k^t)\| \leq L\|\omega_k^{t+1} - \omega_k^t\|. \tag{8}$$

Assumption 3.2. Bounded gradient. Each client's local stochastic gradient variance is bounded, i.e.,

$$\begin{aligned}&\mathbb{E}_{B \subseteq D_k}\|\nabla f^k(\omega_k; \mathrm{B}) - \nabla f^k(\omega_k)\|^2 \leq \sigma^2, \forall k, \forall \omega_k, \\ &\frac{1}{C}\sum_{k=0}^{C-1}\|\nabla f^k(\omega_k) - \nabla F(\omega)\|^2 \leq \mathbb{B}^2, \forall k, \forall \omega_k,\end{aligned} \tag{9}$$

Assumption 3.3. Initializing model weights to 0.

Based on the above assumptions and the principle of SCSS, we denote k-th client's local model update by:

$$\begin{aligned}\omega_{k,s}^{t+1} - \omega_{k,s}^t &= g_{(\cdot,s)}^t W_s^t, \\ g_{(\cdot,s)}^t &:= ((g_{(0,s)}^t)^T, ..., (g_{(k,s)}^t)^T, ..., (g_{(NC-1,s)}^t)^T).\end{aligned} \tag{10}$$

where $\omega_{k,s}^t$ is k-th client's s-th segment in t-th round, and $g_{(k,s)}^t$ is its gradient vector. W_s^t is the weight matrix, and $[W_s^t]_{i,j}$ weights the s-th segment's gradients received by $client_j$ from $client_i$. $[W_s^t]_{i,j} = 0$ means that $client_j$ does not receive s-th segment's gradients from $client_i$, $[W_s^t]_{i,j} = 1/NC$ means all qualified clients upload their local s-th segments to the server. Fig. 4 shows the weight matrices of the above toy example during 0-1 rounds.

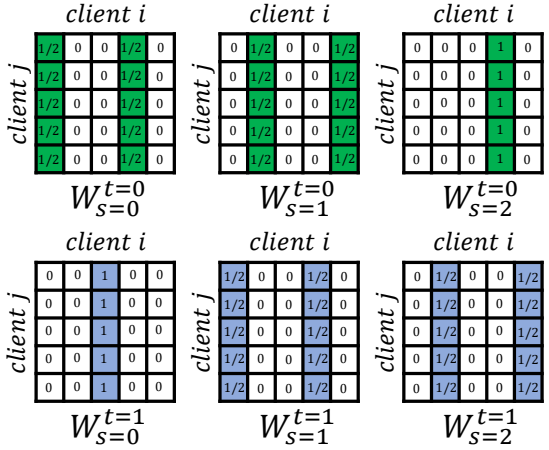


Figure 4: Weight matrices of the toy example in 0-1 rounds.

Yu et al. (2019); Wang et al. (2020) have proved that the following properties are satisfied in the case of randomly discarding a fraction of the gradients/parameters:

$$\begin{aligned}&(\mathbb{E}[W_s^t])A_{NC} = A_{NC},\\ &\mathbb{E}[W_s^t(W_s^t)^T] = \alpha_1 I_{NC} + (1-\alpha_1)A_{NC},\\ &\mathbb{E}[W_s^t A_{NC}(W_s^t)^T] = \alpha_2 I_{NC} + (1-\alpha_2)A_{NC},\end{aligned} \tag{11}$$

where α_1, α_2 are constants and $0 < \alpha_2 < \alpha_1 < 1$; I_{NC} is an identity matrix; A_{NC} is a $NC \times NC$ matrix with all elements as $1/NC$. If each client uploads no duplicate seg_id within a round, then $\alpha_1 = 1$; if all clients upload the full model involving all seg_ids, then $\alpha_1 = 0$.

Since our SCSS algorithm transmits only one specific segment per client per round, $\alpha_1^{s,t}, \alpha_2^{s,t}$ vary with seg_id (s) and round (t). Based on the derived convergence rate of Theorem 1 in Yu et al. (2019); Wang et al. (2020), we replace $\alpha_1^{s,t}, \alpha_2^{s,t}$ with $\alpha_{1max}(\max_{s,t}\alpha_1^{s,t}), \alpha_{2max}(\max_{s,t}\alpha_2^{s,t})$. Hence, we can further derive the following theorem:

Theorem 3.4. *SCSS can converge over wall-to-wall time.*

Proof. Based on Assumption 3.1, we set the learning rate η sufficiently small to satisfy $1 - \frac{6L^2\eta^2}{(1-\sqrt{\beta_{\max}})^2} > 0$, so the convergence rate of SCSS is:

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}(\mathbb{E}\|\nabla f(\bar{\omega}^t)\|^2 + (1-L\eta)\mathbb{E}\|\bar{\nabla} f(\varpi^t)\|^2)\\ &\le \frac{2f(0)-2f(\omega^*)}{\eta T} + \frac{\eta L\sigma^2}{NC} + 4\alpha_{2max}L\eta(\sigma^2+3\mathbb{B}^2)\\ &+ \frac{(2\alpha_{2max}L\eta + L^2\eta^2 + 12\alpha_{2max}L^3\eta^3)\sigma^2 M_1}{(1-\sqrt{\beta_{max}})^2}\\ &+ \frac{3(2\alpha_{2max}L\eta + L^2\eta^2 + 12\alpha_{2max}L^3\eta^3)\mathbb{B}^2 M_1}{(1-\sqrt{\beta_{max}})^2\eta^3)},\end{aligned} \tag{12}$$

where $\nabla f(\bar{\omega}^t) = \nabla f(\frac{n_k}{n}\sum_{k=0}^{NC-1}\omega_k^t)$, $\nabla f(\varpi^t) = \sum_{k=0}^{NC-1}\nabla f^k(\omega_k^t)$, $\beta_{max} = \max_{s,t}(\alpha_1^{s,t} - \alpha_2^{s,t}) < 1$, and $M_1 = (1 - \frac{6L^2\eta^2}{(1-\sqrt{\beta_{\max}})^2})^{-1}$.

When choosing the appropriate learning rate $\eta = \frac{(1-\sqrt{\beta_{max}})^2}{6L+3(\sigma+\mathbb{B})\sqrt{\alpha_{2max}T}+\frac{\sigma\sqrt{T}}{\sqrt{C}}}$, we get:

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\omega}^t)\|^2 \le \frac{(2f(\vec{0})-2f(\omega^*)+L)\sigma}{\sqrt{NC\cdot T}(1-\sqrt{\beta_{max}})}\\ &+ \frac{(2f(\vec{0})-2f(\omega^*)+L)(\sigma+\mathbb{B})}{1-\sqrt{\beta_{max}}}\sqrt{\frac{\alpha_{2max}}{T}}\\ &+ \frac{L^2(\sigma^2+\mathbb{B}^2)}{(\frac{T}{NC}+\alpha_{2max}T)\sigma^2 + \alpha_{2max}T\mathbb{B}^2}\\ &+ \frac{(2f(\vec{0})-2f(\omega^*))L}{T}\end{aligned} \tag{13}$$

From the first main term, we can see that the convergence rate of SCSS is approximately $1/\sqrt{NC\cdot T}$, which is consistent with distributed SGD (Lian et al., 2017). □

3.4. Searching Optimal Hyperparameters

3.4.1. Problem Definition

Based on the $1/\sqrt{NC\cdot T}$ convergence rate, we further make the following assumptions:

Assumption 3.5. Assuming the loss of the global model is bounded, i.e., it satisfies:

$$\mathbb{E}\|f(\vec{0}) - f(\omega^*)\|^2 \le \delta^2. \tag{14}$$

Assumption 3.6. Assuming the global model can reach the maximum convergence rate ε, i.e.,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\omega}^t)\|^2 \le \varepsilon. \tag{15}$$

Lemma 3.7. *The training rounds required by the global model to reach ε convergence rate are:*

$$\begin{aligned}T &= \frac{(2\delta+L)^2\sigma^2}{\varepsilon^2(1-\sqrt{1-\frac{(NC\%NS)+(NC/NS)NC}{NC^2}})^2\alpha C}\\ &= \mathcal{O}(\frac{1}{(1-\sqrt{1-\frac{(\alpha C\%NS)+(\alpha C/NS)\alpha C}{(\alpha C)^2}})^2\alpha C}).\end{aligned} \tag{16}$$

where C is the total number of clients, α is the fraction of sampled clients, $NC = \alpha C$.

We prove lemma 3.7 in Appendix A. Based on lemma 3.7, we can further derive the uplink communication traffic required to reach convergence rate ε:

$$\begin{aligned}\phi(\alpha, NS, p) &= \frac{|\omega|}{NS}(32p+1)\cdot\alpha C\cdot T\\ &= \frac{|\omega|(32p+1)\alpha C\mu}{NS(1-\sqrt{1-\frac{(\alpha C\%NS)+(\alpha C/NS)\alpha C}{(\alpha C)^2}})^2\alpha C}\\ &= \frac{|\omega|(32p+1)\mu}{NS\cdot(1-\sqrt{1-\frac{(\alpha C\%NS)+(\alpha C/NS)}{(\alpha C)^2}})^2\alpha C},\end{aligned} \tag{17}$$

μ is a constant for approximating the big-$\mathcal{O}$ in lemma 3.7.

Definition. We define the purpose of the hyperparameter searching algorithm as: minimizing the communication traffic when reaching ε convergence rate, i.e.,

$$\min \phi(\alpha, NS, p), \tag{18}$$

where $NS \in [1, \alpha C]$, $|\omega|\%NS = 0$; $p \in (0, 1]$, $\alpha \in (0, 1]$.

3.4.2. Problem Solving

Formula (18) is a typical single-objective optimization problem. Since the three independent variables NS, p, α are discrete, this problem can be converted into a combinatorial optimization problem. We exploit *Genetic Algorithm (GA)* to seek the optimal combination of the three hyperparameters iteratively, and the detailed optimal hyperparameter searching algorithm is displayed in Appendix B, Algo. 2.

4. Experiments

We implement QSFL[3] on the FL framework developed in Luo et al. (2019) and use 4 NVIDIA GeForce RTX 3090 GPUs to execute QSFL parallelly. We evaluate QSFL in an image classification task and an object detection task.

4.1. Setup

Model and Dataset. **CNN on FEMINIST**: we train a CNN network (2Conv + 1FC) with 110526 parameters on a real-world FEMINIST[4] dataset (Caldas et al., 2018b). Since the 62-class handwritten characters in FEMINIST have different writing styles, these images show *naturally non-iid* distribution. We assign the images of the first 36 writers to 36 clients. Each client's local dataset is divided into training/testing sets with a ratio of 8:2. **YOLOv3 on Street**: we train a YOLOv3 model with 61556044 parameters on a real-world Street[5] dataset (Luo et al., 2019), which involves 956 images with 7 objects (table, chair, etc.) from 26 streets' cameras. Due to different street locations, the classes and numbers of objects in multiple images are quite different, i.e., *naturally non-iid*. Referring to Luo et al. (2019), we assign a shared testing set to all clients, which involves images from 6 cameras and several images randomly sampled from other cameras, and the unextracted images of the remaining 20 cameras are allocated as training sets to 8 clients.

Baselines. We compare the following methods: a) *Distributed SGD* (Lian et al., 2017); b) Delaying communication: FedAvg (McMahan et al., 2017); c) Sparsification: top-$p\%$ sparsification (Aji et al., 2017), Slim-DP (Sun et al., 2018), HeteroFL (Diao et al., 2021); d) Quantization: signSGD (Bernstein et al., 2018), Deep Compression (Han et al., 2016), C_{nat} (Horvath et al., 2019); e) Combinations of sparsification and quantization: STC (Sattler et al., 2019b), SBC (Sattler et al., 2019a), random subsampling & 2-bit quantization (Caldas et al., 2018a), top-$p\%$ sparsification & signSGD; f) Parameter encoding: SKETCHED-SGD (Ivkin et al., 2019), FetchSGD (Rothchild et al., 2020); g) Client sampling: AFL (Goetz et al., 2019), CMFL (Wang et al., 2019), LAG (Chen et al., 2018a), LAQ (Sun et al., 2019).

Evaluation Metrics. **a) mAP**: we use mean Average Precision (mAP) to evaluate the performance of the global model, it can be computed by $mAP = \frac{\sum_{i=1}^{n} AP_i}{n}$, $AP_i = \frac{1}{m_i}\sum_{j=1}^{m_i} \mathbb{I}(f(x_j) = y_j)$, where AP is the average prediction accuracy of each class, n and m_i are the total number of classes and instances, $\mathbb{I}(\cdot)$ is an indicator function. **b) Compression Ratio**: *Distributed SGD* requires R_0 rounds to attain the target accuracy, hence consuming $32|\omega|R_0C$ bits in uplink communication. The compression ratio can be computed by (the communication traffic of algorithms) / $32|\omega|R_0C$ when reaching the *same target accuracy*. All experimental results are measured only for the uplink communication after convergence.

[3]https://github.com/LipingYi/QSFL

[4]https://github.com/TalwalkarLab/leaf/tree/master/data/FEMINIST

[5]https://dataset.fedai.org

Table 1: Results of QSFL and baselines on FEMINIST. We execute multiple experiments for each algorithm with different hyperparameters, and compute compression ratio when algorithms reach 70% target mAP (most of them can reach it). We report two typical cases: a) When reaching maximum compression ratios (column-3), the mAP reached in the last (200-th) round (column-4). b) When reaching maximum mAP (column-5), compression ratio achieved by algorithms (column-6). '-': not reaching 70% mAP.

CNN on FEMINIST, 200 Rounds					
Type	**Algorithm**	**max_CR (70%)**	**mAP (200)**	**max_mAP (200)**	**CR (70%)**
No Optimization	Distributed SGD	1.00×	78.87%	78.87%	1.00×
Communication Delay	FedAvg (E=10,B=1)	19.00×	**86.54%**	86.54%	19.00×
Quantization	signSGD	32.00×	70.09%	70.09%	32.00×
	Deep Compression	-	8.43%	8.43%	-
	C_{nat}	86.86×	75.31%	75.31%	86.86×
Sparsification	Top-p (%)	95.00×	75.55%	85.93%	10.56×
	Slim-DP	54.29×	79.26%	85.42%	13.57×
	HeteroFL	85.30×	76.25%	86.07%	10.35×
Combination of Quantization and Sparsification	Top-p (%) & signSGD	368.48×	70.73%	70.74%	10.24×
	Random subsample	-	8.67%	8.67%	-
	SBC	-	6.07%	6.07%	-
	STC	-	66.41%	66.41%	-
Client Sampling	AFL	126.67×	82.42%	84.85%	38.00×
	CMFL	42.22×	77.65%	83.60%	30.40×
	LAG	19.00×	84.30%	84.30%	19.00×
	LAQ	32.33×	72.25%	82.73%	24.42×
Parameter Encoding	SKETCHED-SGD	81.25×	74.28%	85.12%	10.15×
	FetchSGD	237.52×	72.35%	83.65%	4 1.50×
Ours	**QSFL**	**889.76**×	78.08%	**86.63%**	**157.79**×

Table 2: Results of QSFL and baselines on Street dataset.

YOLOv3 on Street, 300 Rounds					
Type	**Algorithm**	**max_CR (70%)**	**mAP (300)**	**max_mAP (300)**	**CR (70%)**
No Optimization	Distributed SGD	1.00×	**84.00%**	84.00%	1.00×
Communication Delay	FedAvg (E=5,B=1)	1.49×	83.20%	84.20%	1.24×
Quantization	signSGD	**32.00**×	10.60%	10.60%	**32.00**×
	C_{nat}	-	69.30%	75.20%	0.84×
Sparsification	Top-p (%)	1.08×	81.60%	81.60%	1.08×
	HeteroFL	1.05×	82.20%	82.20%	1.05×
Combination of Quantization and Sparsification	Top-p (%) & signSGD	-	8.20%	9.50%	-
Client Sampling	AFL	1.49×	83.20%	84.10%	1.37×
	CMFL	1.49×	83.20%	84.00%	1.31×
	LAG	1.15×	81.08%	82.79%	1.13×
	LAQ	1.32×	79.22%	81.32%	1.04×
Parameter Encoding	SKETCHED-SGD	1.03×	82.28%	82.28%	1.03×
	FetchSGD	1.58×	76.65%	80.60%	1.42×
Ours	**QSFL**	6.35×	83.21%	**84.50%**	3.58×

4.2. Experimental Results

4.2.1. Comparisons with Baselines

Tab. 1 and Tab. 2 record the results of all algorithms when respectively achieving the highest compression ratio and highest mAP in two tasks. From Tab. 1, we can see that *QSFL owns the highest compression ratio (*$889.76\times$*) and also has a similar mAP with Distributed SGD. Besides, QSFL achieves the highest mAP (*86.63%*) while also maintaining the highest compression ratio (*$157.786\times$*). QSFL also outperforms alternatives in the object detection task.*

We also choose the above each type of algorithms which perform best in the two cases of Tab. 8, and report how the loss and mAP range over rounds in Appendix D, Fig. 9. In the case of achieving maximum compression ratio, as shown in Fig. 9 (a-b), we can see: a) Owing to the non-iid distribution, distributed SGD shows the worst convergence rate. b) QSFL sharply shakes in the early training stage due to only uploading one segment per client. But it finally converges to a better mAP than most baselines. c) Although FedAvg achieves better mAP than QSFL, it only achieves $19\times$ compression ratio while QSFL reaches $889.76\times$. In the case of achieving the maximum mAP shown in Fig. 9 (c-d), we can see QSFL has a stable convergence rate and performs similar model accuracy with FedAvg while having $8.3\times$ higher compression ratio. Both of the two typical cases display that *QSFL converges over wall-to-wall time, which is consistent with the convergence analysis in Sec. 3.3.*

4.2.2. Performance Evaluation of QSFL

(A) Searching Optimal (E, B) for FedAvg. We first search optimal (E, B) for FedAvg to maintain minimum communication cost. From preliminary experiments recorded in Appendix C, we find *FedAvg (E = 10, B = 1)* on FEMINIST achieves the highest $19\times$ compression ratio with 86.54% mAP, *FedAvg (E = 5, B = 1)* on Street attains the highest $1.49\times$ compression ratio with 83.23% mAP, which are the default (E, B) settings in subsequent experiments. We also report other hyperparameters of FL in Appendix C.

(B) Results of QJ. From Tab. 3, we can see that mAP decreases, and the compression ratio rises as α (the ratio of qualified clients) decreases for each fixed β (hyperparameter of QJ), which is reasonable since fewer qualified clients lead to less local information sent to the server. Besides, for each fixed α, different values of β almost achieve similar compression ratios. *β = 0.9 in both tasks keeps relatively high mAP and compression ratio*. So we can conclude that removing redundant clients by QJ keeps the model performance while improving communication efficiency.

(C) Results of SCSS. We evaluate SCSS with different (NS, p) in two tasks. On the FEMINIST dataset, 36 clients join in FL, so $NS \in [1, 36]$. Furthermore, we take the factors $(1, 2, 6, 13, 26)$ of 110526 (the CNN model's parameter capacity) as the values of NS for uniform segmentation. On the Street dataset, 8 clients participate in FL, so $NS \in [1, 8]$. We also choose the factors $(1, 2, 4)$ of 61556044 (the YOLOv3 model's parameter capacity) for NS.

From Fig. 5 (a-b), we can see: a) for fixed NS, decreasing p makes mAP drop and compression ratio increase; b) for fixed p, mAP drops and compression ratio increases as NS grows since shorter segment carries less information. *NS = 6, 13 achieve better trade-offs*; c) $p = 0.3$ is an approximate inflection point of four curves, i.e., mAP and compression ratio change more smoothly when $p \geq 0.3$ and otherwise show significant fluctuations; d) the compression ratio of $p = 0.05$ is less than $p = 0.1$, because too low p makes the uploaded parameters carry little information and requires more rounds to achieve 70% target mAP. So *p = [0.1, 0.3] takes good trade-offs between mAP and compression ratio*.

Table 3: Results of QJ. mAP (200/300) means the mAP in the 200/300-th round, CR (70%) denotes the compression ration measured in 70% target mAP.

CNN on FEMINIST (200 rounds)				YOLOv3 on Street (300 rounds)			
α	β	mAP (200)	CR (70%)	α	β	mAP (300)	CR (70%)
0.9	0.9	85.28%	28.15×	0.9	0.9	85.7%	**1.71**×
0.7		84.48%	36.19×	0.7		79.5%	0.68×
0.5		**85.72%**	38.00×	0.5		79.9%	0.75×
0.3		85.42%	76.00×	0.9	0.8	84.5%	1.30×
0.1		85.34%	95.00×	0.7		85.1%	0.89×
0.05		84.82%	80.00×	0.5		76.5%	0.91×
0.5	0.8	85.07%	38.00×	0.9	0.7	**86.8%**	1.40×
0.1		84.95%	95.00×	0.7		83.9%	1.00×
0.05		83.62%	**116.92**×	0.5		78.6%	0.84×
0.5	0.7	85.10%	38.00×	0.9	0.6	83.9%	1.57×
0.1		84.27%	108.57×	0.7		86.7%	1.21×
0.05		82.86%	76.00×	0.5		80.3%	0.83×
0.5	0.6	84.45%	30.40×	0.9	0.5	85.7%	1.47×
0.1		83.67%	69.09×	0.7		84.1%	0.84×
0.05		82.19%	89.41×	0.5		83.6%	1.27×
0.5	0.5	85.02%	30.40×	0.9	0.1	86.2%	1.19×
0.1		83.97%	84.44×	0.7		70.9%	0.68×
0.05		82.91%	76.00×	0.5		61.9%	-

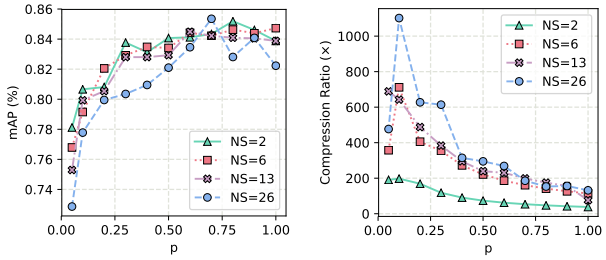
Fig. 5 (c-d) display that *(NS = 2, p = 0.5) achieves both great mAP and compression ratio on the Street dataset*.

(D) Results of QSFL. Based on the conclusions of **(B)**, we evaluate QSFL with QJ ($\beta = 0.9$) in both two tasks.

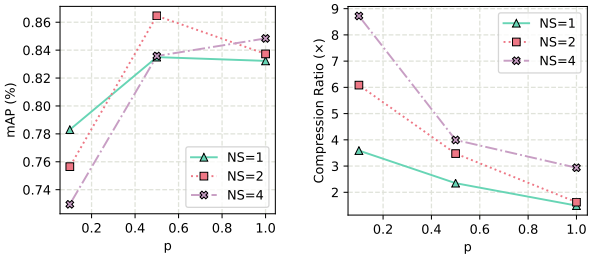
On FEMINIST dataset, we find that $NS = 13$ (64.92%) doesn't attain 70% target mAP even if $\alpha = 0.9$ and $p = 0.9$ in 200 rounds. Also considering optimal values (6,13) of NS concluded in **(C)**, we evaluate QSFL with $NS = 6$. From Fig. 6 (a), we can see that mAP changes smoothly when $\alpha \geq 0.7$ and otherwise drops quickly as α decreases for fixed p. Since the settings of $\alpha \leq 0.3$ with any p do not achieve 70% target mAP, so these points are not displayed in Fig. 6 (b). Fig. 6 (b) shows that the settings of any fixed p except $p \leq 0.3$ performs higher compression ratio as α drops. *Especially, (p = 0.1, α = 0.5) generate the highest compression ratio ($889.76\times$) with 78.08% mAP which is only 0.79% lower than Distributed SGD (78.87%).* Fig. 6 (c-d) indicate that decreasing p slightly affects mAP but effectively improves the compression ratio for fixed α. Therefore, *QSFL (β = 0.9, NS = 6, α = 0.7, p = 0.4) makes the best trade-off between communication and accuracy, it achieves 311× compression ratio with 5.24% mAP rise*.

On the Street dataset, we also test QSFL with $\alpha \in [0.1, 0.9]$ and $p \in [0.1, 0.9]$ when $NS = 2$, and find that *QSFL (β = 0.9, NS = 2, α = 0.9, p = 0.5) achieves highest $6.35\times$ compression ratio and 83.21% mAP (Distributed SGD: 84%)*.

As shown in Appendix B, the above (α^*, NS^*, p^*) in two tasks are both close to the searching results of Algo. 2.

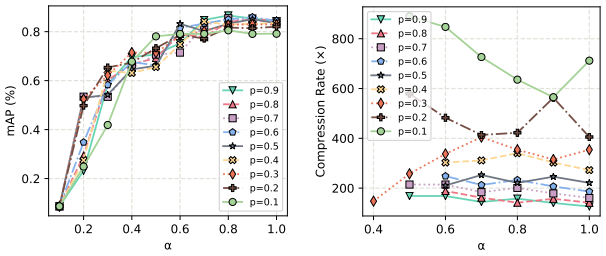


(a) mAP v.s. p (b) CR v.s. p

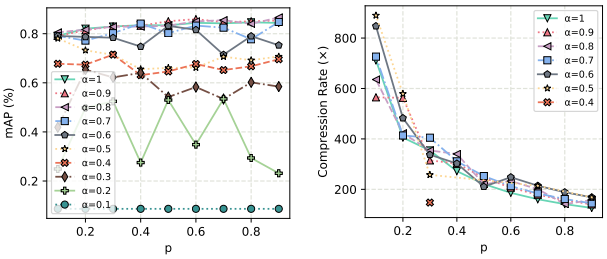


(c) mAP v.s. p (d) CR v.s. p

Figure 5: (a-b) display the results of SCSS on FEMINIST dataset, (c-d) show the results on Street dataset.



(a) mAP v.s. α (b) CR v.s. α



(c) mAP v.s. p (d) CR v.s. p

Figure 6: Effects of QSFL vary with client sampling rate α and sparsification rate top-p on the FEMINIST dataset.

4.2.3. QSFL WITH MORE CLIENTS

When more clients join in FL, segments with any id are provided by more clients, in which more enough information contributes to aggregation and the convergence rate will be improved. To verify this conjecture, we test $C \in \{10, 20, 30, 40, 50\}$ clients on the FEMINIST dataset. We also configure QSFL with ($\beta = 0.9$, $NS = 6$, $\alpha = 0.7$, $p = 0.4$), and the results are presented in Appendix F, Fig. 10.

As shown in Fig. 10 (a), we find that QSFL presents oscillation when the number of clients is less than 40, otherwise it stably converges over wall-to-wall time, which verifies our conjecture. To clarify oscillation, we test SCSS (i.e., QJ ($\alpha = 1$)) with the above settings. As shown in Fig. 10 (b), oscillation disappears in all settings, which indicates that the oscillation is probably caused by QJ sampling quality-poor clients in some rounds. SCSS shows robustness to number of clients, since it guarantees that any segment of the global model is provided by at least one client.

4.2.4. ABLATION EXPERIMENTS

We conduct ablation experiments to test the importance of each part in QSFL. From Tab. 8 in Appendix E, we can see that the combination of any two-part has a higher compression ratio than any single part, and the combination of three parts shows the best compression ratio, which indicates both QJ and SCSS promote a communication-efficient QSFL.

Summary. Extensive experiments verify that QSFL takes a great trade-off between compression ratio and model accuracy, so users can control its hyperparameters for varying requirements of communication cost and model effect.

Broader Impact. QJ and SCSS can be used as two independent approaches to reduce FL uplink communication costs. QSFL with QJ and SCSS is applicable in a *cross-device* FL scenario *with numerous participating clients*, while QSFL with only SCSS can be used in a *cross-device* or *cross-silo* FL scenario *without limitation of number of clients*.

5. Concluding Remarks and Future Work

In this paper, the proposed QSFL framework optimizes FL uplink communication overhead from two levels: rejecting redundant clients uploading model updates (client-level) and compressing models to unique segments for uploading (model-level). Theoretical proof and extensive experiments verify that QSFL can effectively reduce uplink communication costs with marginal model accuracy degradation. In future work, (a) we will explore an effective *downlink* communication optimization solution to couple with QSFL's existing superiority in *uplink* communication, so as to conduct a two-way communication-efficient FL framework. Besides, (b) we will also develop current SCSS with *dynamic segments*, which allows each client to upload one segment with size adaptive to system resources. In this way, QSFL's robustness to heterogeneous devices will be enhanced.

Acknowledgements

This research is supported in part by the National Science Foundation of China under Grant 62141412 and Grant 61872201, in part by the Science and Technology Development Plan of Tianjin under Grant 20JCZDJC00610 and Grant 19YFZCSF00900, in part by the Key Research and Development Program of Guangdong under Grant 2021B0101310002, and in part by the Fundamental Research Funds for the Central Universities. The authors thank the advice given by reviewers for improving this work, and also thank Heng Zhang and Rui Zhang for their feedback.

References

Aji, A. F., Heafield, K., , and . Sparse communication for distributed gradient descent. In *Proc. EMNLP*, pp. 440–445, Copenhagen, Denmark, 2017. ACL.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: communication-efficient SGD via gradient quantization and encoding. In *Proc. NeurIPS*, pp. 1709–1720, Long Beach, CA, USA, 2017.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. SIGNSGD: compressed optimisation for non-convex problems. In *Proc. ICML*, volume 80, pp. 559–568, Stockholm, Sweden, 2018. PMLR.

Cai, W., Li, W., , and . Weight normalization based quantization for deep neural network compression. *CoRR*, abs/1907.00593, 2019.

Caldas, S., Konečný, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *CoRR*, abs/1812.07210, 2018a.

Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H. B., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018b.

Chen, M., Shlezinger, N., Poor, H. V., Eldar, Y. C., and Cui, S. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17), 2021.

Chen, T., Giannakis, G. B., Sun, T., and Yin, W. LAG: lazily aggregated gradient for communication-efficient distributed learning. In *Proc. NeurIPS 2018, Montréal, Canada*, pp. 5055–5065, 2018a.

Chen, Z., Wang, S., Wu, D. O., Huang, T., and Duan, L. From data to knowledge: Deep learning model compression, transmission and communication. In *Proc. MM*, pp. 1625–1633, Seoul, Republic of Korea, 2018b. ACM.

Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *Proc. ICLR*. OpenReview.net, 2021.

Goetz, J., Malik, K., Bui, D., Moon, S., Liu, H., and Kumar, A. Active federated learning. *CoRR*, abs/1909.12641, 2019.

Guo, Y., Yao, A., , and . Network sketching: Exploiting binary structure in deep cnns. In *Proc. CVPR*, pp. 4040–4048, Honolulu, HI, USA, 2017. IEEE Computer Society.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proc. ICLR 2016*, 2016.

Horvath, S., Ho, C., Horvath, L., Sahu, A. N., Canini, M., and Richtárik, P. Natural compression for distributed deep learning. *CoRR*, abs/1905.10988, 2019.

Hu, C., Jiang, J., and Wang, Z. Decentralized federated learning: A segmented gossip approach. *CoRR*, abs/1908.07782, 2019.

Ivkin, N., Rothchild, D., Ullah, E., Braverman, V., Stoica, I., and Arora, R. Communication-efficient distributed SGD with sketching. In *Proc. NeurIPS*, pp. 13144–13154, Vancouver, BC, Canada, 2019.

Kairouz, P., Kairouz, P., McMahan, H. B., Avent, B., and et al. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. In *Proc. NeurIPS, Workshop*, 2016.

Leng, C., Dou, Z., Li, H., Zhu, S., and Jin, R. Extremely low bit neural network: Squeeze the last bit out with ADMM. In *Proc. AAAI*, pp. 3466–3473, USA, 2018. AAAI Press.

Li, T., Liu, Z., Sekar, V., and Smith, V. Privacy for free: Communication-efficient learning with differential privacy using sketches. *CoRR*, abs/1911.00972, 2019.

Lian, X., Zhang, C., Zhang, H., Hsieh, C., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Proc. NIPS, Long Beach, CA, USA*, pp. 5330–5340, 2017.

Lin, X., Zhao, C., Pan, W., and . Towards accurate binary convolutional neural network. In *Proc. NeurIPS*, pp. 345–353, Long Beach, CA, USA, 2017.

Luo, J., Wu, X., Luo, Y., Huang, A., Huang, Y., Liu, Y., and Yang, Q. Real-world image datasets for federated learning. *CoRR*, abs/1910.11089:1–8, 2019.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proc. AISTATS*, volume 54, pp. 1273–1282, Fort Lauderdale, FL, USA, 2017. PMLR.

Reagen, B., Gupta, U., Adolf, B., Mitzenmacher, M., Rush, A. M., Wei, G., and Brooks, D. Weightless: Lossy weight encoding for deep neural network compression. In *Proc.*

ICML, volume 80, pp. 4321–4330, Stockholmsmässan, Stockholm, Sweden, 2018. PMLR.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *Proc. AISTATS*, volume 108, pp. 2021–2031, Online, 2020. PMLR.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd: Communication-efficient federated learning with sketching. In *Proc. ICML*, volume 119, pp. 8253–8265, Virtual, 2020. PMLR.

Sattler, F., Wiedemann, S., Müller, K., and Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. In *Proc. IJCNN*, pp. 1–8, Budapest, Hungary, 2019a. IEEE.

Sattler, F., Wiedemann, S., Müller, K., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *CoRR*, abs/1903.02891, 2019b.

Strom, N. Scalable distributed DNN training using commodity GPU cloud computing. In *Proc. INTERSPEECH*, pp. 1488–1492, Dresden, Germany, 2015. ISCA.

Sun, J., Chen, T., Giannakis, G. B., and Yang, Z. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Proc. NeurIPS, Vancouver, BC, Canada*, pp. 3365–3375, 2019.

Sun, S., Chen, W., Bian, J., Liu, X., and Liu, T. Slim-dp: A multi-agent system for communication-efficient distributed deep learning. In *Proc. AAMAS*, pp. 721–729, Stockholm, Sweden, 2018.

Tsuzuku, Y., Imachi, H., Akiba, T., and . Variance-based gradient compression for efficient distributed deep learning. In *Proc. ICLR*, Vancouver, BC, Canada, 2018. OpenReview.net.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Proc. NeurIPS, Vancouver, BC, Canada*, pp. 14236–14245, 2019.

Wang, H., Sievert, S., Liu, S., Charles, Z. B., Papailiopoulos, D. S., and Wright, S. J. ATOMO: communication-efficient learning via atomic sparsification. In *Proc. NeurIPS*, pp. 9872–9883, Montréal, Canada, 2018.

Wang, H., Chen, J., Wan, X., Tian, H., Xia, J., Zeng, G., Wang, W., Chen, K., Bai, W., and Jiang, J. Domain-specific communication optimization for distributed DNN training. *CoRR*, abs/2008.08445, 2020.

Wang, L., Wang, W., Li, B., and . CMFL: mitigating communication overhead for federated learning. In *Proc. ICDCS*, pp. 954–964, Dallas, TX, USA, 2019. IEEE.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Proc. NeurIPS*, pp. 1306–1316, Montréal, Canada, 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Proc. NeurIPS*, pp. 1509–1519, Long Beach, CA, USA, 2017.

Xu, J., Du, W., Cheng, R., He, W., and Jin, Y. Ternary compression for communication-efficient federated learning. *CoRR*, abs/2003.03564, 2020.

Yao, X., Huang, T., Wu, C., Zhang, R., and Sun, L. Towards faster and better federated learning: A feature fusion approach. In *Proc. ICIP 2019, Taipei, China*, pp. 175–179. IEEE, 2019.

Yi, L., Zhang, J., Zhang, R., Shi, J., Wang, G., and Liu, X. Su-net: an efficient encoder-decoder model of federated learning for brain tumor segmentation. In *Proc. ICANN*, pp. 761–773. Springer, 2020.

Yu, C., Tang, H., Renggli, C., Kassing, S., Singla, A., Alistarh, D., Zhang, C., and Liu, J. Distributed learning over unreliable networks. In *Proc. ICML, Long Beach, California, USA*, volume 97, pp. 7202–7212. PMLR, 2019.

Zhang, H., Wang, X., Chen, J., Wang, C., and Li, J. D2d-lstm: Lstm-based path prediction of content diffusion tree in device-to-device social networks. In *Proc. AAAI*, volume 34, pp. 295–302, 2020.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

A. Proof for Lemma 3.7

Proof. According to the first main term on the right-hand side of the inequality in Eq. 13 and Assumptions 3.5 and 3.6, let

$$\begin{aligned}\varepsilon &= \frac{(2f(\vec{0}) - 2f(\omega^*) + L)\sigma}{\sqrt{NC \cdot T}(1 - \sqrt{\beta_{max}})} \\ &= \frac{(2\delta + L)\sigma}{\sqrt{NC \cdot T}(1 - \sqrt{\beta_{max}})},\end{aligned} \tag{19}$$

where $NC = \alpha C$. Then we can further get:

$$T = \frac{(2\delta + L)^2\sigma^2}{\varepsilon^2(1 - \sqrt{\beta_{max}})^2\alpha C}. \tag{20}$$

We refer to the approximation of β_{max} in Yu et al. (2019), i.e., the mean drop probability of segments. We first derive the mean uploading probability of segments:

$$\begin{aligned}P_{up} &= \frac{NC\%NS}{NC} \cdot \frac{(NC/NS) + 1}{NC} \\ &+ (1 - \frac{NC\%NS}{NC}) \cdot \frac{NC/NS}{NC} \\ &= \frac{(NC\%NS) + (NC/NS)NC}{NC^2}.\end{aligned} \tag{21}$$

We show two examples to help understand how the above formula works, as shown in Fig. 7.

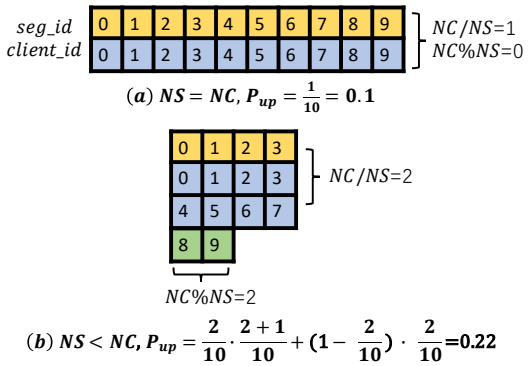


Figure 7: Examples of computing uploading probability.

So we can get $\beta_{max} = 1 - P_{up}$, i.e.,

$$\beta_{max} = 1 - P_{up} = 1 - \frac{(NC\%NS) + (NC/NS)NC}{NC^2}. \tag{22}$$

Substituting β_{max} of formula 19 with formula 22,

$$\begin{aligned}T &= \frac{(2\delta + L)^2\sigma^2}{\varepsilon^2(1 - \sqrt{1 - \frac{(NC\%NS) + (NC/NS)NC}{NC^2}})^2\alpha C} \\ &= \mathcal{O}(\frac{1}{(1 - \sqrt{1 - \frac{(\alpha C\%NS) + (\alpha C/NS)\alpha C}{(\alpha C)^2}})^2\alpha C}).\end{aligned} \tag{23}$$

□

B. Optimal Parameter Searching Algorithm

The following algorithm is the optimal hyperparameter searching algorithm.

Algorithm 2 Optimal Hyperparameter Searching Algorithm

Input: C, the total number of clients; $|\omega|$, the number of model parameters; μ, approximation term
output: NS^*, p^*, α^*
// Setting bounds with the constrains in formula (18)
$bound^{up} = [NS^{ub}, p^{ub}, \alpha^{ub}]$
$bound_{low} = [NS_{lb}, p_{lb}, \alpha_{lb}]$
$\phi(NS, p, \alpha) \leftarrow$ according to formula (17)
ga = GA($\phi(NS, p, \alpha)$, dim=3, $bound_{low}$, $bound^{up}$)
$\vec{x}^*$, y^* = ga.run()
Return $\vec{x}^* = [NS^*, p^*, \alpha^*]$

We execute the above optimal parameter searching algorithm in two tasks respectively and report the actual iterations in Fig. 8. It's obvious to observe that our searching algorithm can converge to the stable and optimal solution.

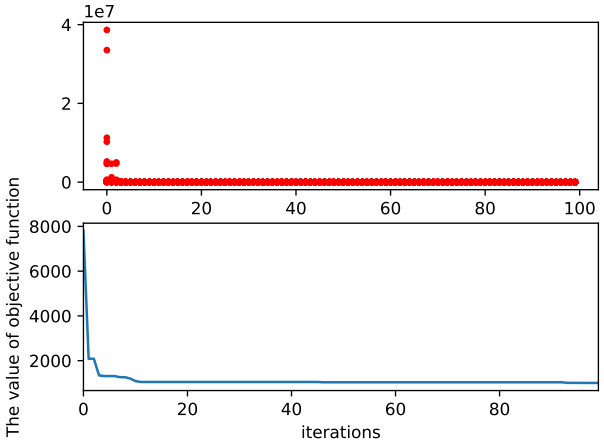


Figure 8: Practical convergence of the optimal parameter searching algorithm.

We report hyperparameters found by the above searching algorithm in Tab. 4. We can see that the searching results are close to the optimal parameters evaluated in experiments.

Table 4: Optimal hyperparameters found by the searching algorithm and experiments.

Task	Experimental Results			Searching Results		
	NS	p	α	NS	p	α
CNN on FEMINIST	6	0.4	0.7	5.71	0.43	0.70
YOLOv3 on Street	2	0.5	0.9	1.94	0.45	0.86

C. Results of Searching Optimal FedAvg(E,B)

Limited to memory (24G) of one NVIDIA GeForce RTX 3090 GPU, B (batch size) is set up to 20 in the image classification task and up to 1 in the object detection task. The results of two tasks are reported in Tab. 5 and Tab. 6 respectively. We find that $(E = 10, B = 1)$ on FEMINIST dataset and $(E = 5, B = 1)$ on Street dataset take better trader-off between compression ratio and model accuracy.

Table 5: FedAvg with different (E,B) in the image classification task. Rounds (70%): required rounds when reaching 70% target mAP, CR (70%): compression ratio when reaching 70% target mAP, mAP (200): mAP in the 200-th round.

FedAvg (E, B), CNN on FEMINIST				
E	B	Rounds (70%)	CR (70%)	mAP (200)
1	20	76	1.00×	78.87%
1	10	24	3.17×	84.97%
1	1	24	3.17×	84.97%
10	20	12	6.33×	85.24%
10	10	8	9.50×	83.31%
10	**1**	**4**	**19.00×**	**86.54%**
20	20	8	9.50×	84.79%
20	10	6	12.67×	84.98%
20	1	**4**	**19.00×**	84.40%

Table 6: FedAvg with different (E, B) in the object detection task. Rounds (70%): required rounds when reaching 70% target mAP, CR (70%): compression ratio when reaching 70% target mAP, mAP (300): mAP in the 300-th round.

FedAvg(E,B), YOLOv3 on Street				
E	B	Rounds (70%)	CR (70%)	mAP (300)
1	1	103	1.00×	**84.00%**
5	**1**	69	**1.49×**	83.23%
10	1	74	0.93×	82.70%

We also report detailed hyperparameters settings of FL in the two tasks, as shown in Tab. 7.

Table 7: Hyperparameters settings of FL. C: total number of clients, η: learning rate; E: epoch, B: batch size.

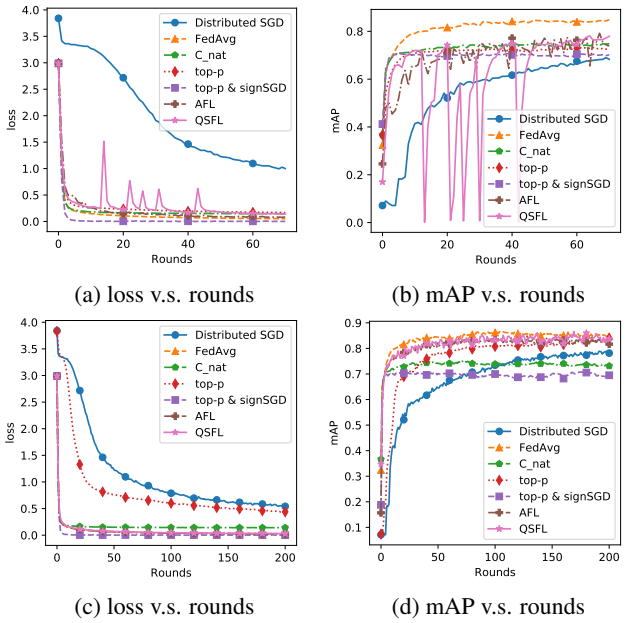
CNN on FEMINIST				YOLOv3 on Street			
C	η	E	B	C	η	E	B
36	0.01	10	1	8	0.01	5	1

D. Experimental Results of Convergence

Experimental results reported in Fig. 9 show QSFL can converge over wall-to-wall time.

E. Results of Ablation Experiments

We show detailed results of ablation experiments in Tab. 8.



(a) loss v.s. rounds (b) mAP v.s. rounds

(c) loss v.s. rounds (d) mAP v.s. rounds

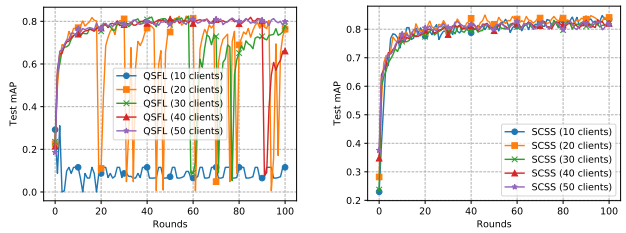
Figure 9: The training loss and test mAP of Six types of advanced communication-efficient algorithms and QSFL vary with rounds in the two typical cases of Tab. 1.

Table 8: Results of ablation experiments with QSFL (β = 0.9, NS = 6, α = 0.5, p = 0.1) on FEMINIST dataset and QSFL (β = 0.9, NS = 2, α = 0.9, p = 0.5) on Street dataset.

QJ	SCSS	SCSS	**FEMINIST**		**Street**	
@α	@NS	@p	CR (70%)	mAP (200)	CR (70%)	mAP (300)
✓	×	×	38.00×	**85.72%**	1.71×	**85.70%**
×	✓	×	57.00×	84.73%	1.62×	83.73%
×	×	✓	63.33×	78.98%	2.35×	83.50%
✓	✓	×	130.29×	74.04%	2.54×	83.29%
✓	×	✓	197.72×	79.01%	2.95×	84.69%
×	✓	✓	593.17×	79.01%	3.48×	83.46%
✓	✓	✓	**889.76×**	78.08%	**6.35×**	83.21%

F. Results of QSFL with More Clients

Here we report the results of QSFL and SCSS with different number of participating clients on the FEMINIST dataset.



(a) (b)

Figure 10: The effects of QSFL and SCSS vary with the number of participating clients on the FEMINIST dataset.