Fed-QSSL: A Framework for Personalized Federated Learning under Bitwidth and Data Heterogeneity

Yiyue Chen[1], **Haris Vikalo**[1], **Chianing Wang**[2] *

[1]The University of Texas at Austin
[2]Toyota InfoTech Lab USA
yiyuechen@utexas.edu, hvikalo@utexas.edu, johnny.wang@toyota.com

Abstract

Motivated by high resource costs of centralized machine learning schemes as well as data privacy concerns, federated learning (FL) emerged as an efficient alternative that relies on aggregating locally trained models rather than collecting clients' potentially private data. In practice, available resources and data distributions vary from one client to another, creating an inherent system heterogeneity that leads to deterioration of the performance of conventional FL algorithms. In this work, we present a federated quantization-based self-supervised learning scheme (Fed-QSSL) designed to address heterogeneity in FL systems. At clients' side, to tackle data heterogeneity we leverage distributed self-supervised learning while utilizing low-bit quantization to satisfy constraints imposed by local infrastructure and limited communication resources. At server's side, Fed-QSSL deploys de-quantization, weighted aggregation and re-quantization, ultimately creating models personalized to both data distribution as well as specific infrastructure of each client's device. We validated the proposed algorithm on real world datasets, demonstrating its efficacy, and theoretically analyzed impact of low-bit training on the convergence and robustness of the learned models.

Introduction

Federated learning (FL) (McMahan et al. 2017; Kairouz et al. 2021) recently emerged in response to the challenges in data privacy, storage cost and computation commonly faced by the conventional (centralized) learning systems. In centralized learning, a server collects and stores data to be used for model training, raising concerns regarding potential leakage of sensitive information and inefficient utilization of the storage and computation resources. Lately, advancements in software and hardware technologies have equipped smart edge devices with computational power that enables local data processing, allowing implementation of distributed frameworks such as FL in a variety of real-world applications (Chen, Sun, and Jin 2019). In FL systems, participating devices collaboratively learn a model while preserving privacy by training on local data that remains private. A server aggregates local updates to obtain a global model which is then distributed to the clients; in turn, the clients continue local training using the latest global model as a starting point. In addition to the conventional supervised learning, federated learning is suitable for meta-learning and unsupervised learning problems (Jiang et al. 2019; Servetnyk, Fung, and Han 2020).

Heterogeneity in local data distribution, as well as imbalance in the computation and memory capabilities of the participating devices, present major challenges to federated learning (Zhao et al. 2018; Yoon et al. 2022). Practical scenarios where both sources of heterogeneity occur include healthcare disorder prediction for patients with different profiles and monitoring devices, transportation systems over different vehicles and terrain, and indoor/outdoor air quality detection, to name just a few. Several recent FL techniques aim to alleviate the impact of data, model or device heterogeneity (Jiang, Shan, and Zhang 2020; Lin et al. 2020; Diao, Ding, and Tarokh 2020; Wang et al. 2022). However, variations in bitwidth available to the clients that participate in training have been much less studied (Yoon et al. 2022). To our knowledge, there exists no prior work in literature investigating FL in setting where both the local data and device bitwidth are heterogeneous – the focus of our work.

On one hand, clients' devices collect and/or generate data at different times and locations, leading to discrepancy in data amounts and distribution. When distributed devices train locally, each learned model optimizes an objective specified in regard to a local dataset. If the data across participating devices is non-IID, the devices end up optimizing distinct objectives which generally adversely affects training convergence and, ultimately, negatively impacts performance of the resulting global model trained by classical FL algorithms such as FedAvg (McMahan et al. 2017; Li et al. 2019). This is exacerbated when a client's dataset contains only a small subset of classes or few data points, leading to an insufficiently expressive local model. In turn, using such models in the aggregation step at the server leads to a global model that lacks robustness and in general may significantly underperform its centrally trained counterparts (Zhao et al. 2018).

On the other hand, to satisfy the constraints on local compute and memory footprint as well as on the communication bandwidth between clients and the central server, each client needs to train a local model in low bitwidth operations and store the updated model in low bitwidth (Yoon et al. 2022). When the clients' devices have different bitwidth capabilities, a number of novel challenges arise including: (1) Quantized

*This work was supported by Toyota Motor North America.

training conducted at lower precision levels does not necessarily lead to expressive local model; (2) the server needs to aggregate local models that are represented by different number of bits; and (3) following the aggregation of local updates, the server should communicate to clients the new global model re-quantized at levels matching the capabilities of different devices. Prior works that deploy model compression in distributed settings focus on reducing the communication cost (Reisizadeh et al. 2020; Chen, Hashemi, and Vikalo 2021a,b); those methods utilize full precision weights in the training process rather than trying to train under weight precision constraint. To address the problem of aggregating models trained at different precision, (Yoon et al. 2022) proposed a progressive weight de-quantizer that transforms low precision models to their full precision versions before aggregating them. The de-quantizer there requires the server to train DenseNet for de-quantizing low precision models; the method assumes the local data distribution is the same across different clients and thus does not apply to the heterogeneous data settings studied in this paper. Note that all of these methods are restricted to supervised learning.

The contributions of this paper are summarized as follows:

1. For the first time, a novel FL problem characterized by heterogeneous local data distributions and varied capabilities of clients' devices is studied. This is the first work to consider the non-trivial task of aggregating models which due to varied bitwidth and heterogeneous data have incompatible parameters and different expressiveness. While prior work addresses either source of heterogeneity in isolation of the other, to our knowledge the combination of the two has not been investigated previously. Indeed, new challenges set forth by the combination of the heterogeneity sources render the prior techniques ineffective.
2. We present a new FL framework, Fed-QSSL, that enables learning personalized and robust low-bitwidth models in settings characterized by diverse infrastructure and data distributions. Fed-QSSL combines low-bitwidth quantization training with self-supervised learning at the client side, and deploys de-quantization, weighted aggregation and re-quantization at the server side.
3. We theoretically analyze the impact of low-bit training on the convergence and robustness of learned models. In particular, we present a bound on the variance of the quantization errors in local and global training and investigate the associated convergence speed. The analysis demonstrates that low-bit training with Fed-QSSL on heterogeneous data allows learning meaningful representations.
4. Finally, the efficacy of the proposed algorithm is tested in a variety of experiments.

Related Work

To address local infrastructure constraints in FL, (Diao, Ding, and Tarokh 2020) rely on models with simple architectures and low-bit parameter representation. However, existing prior work typically assumes that clients train models of same precision, e.g., the same bitwidth, which may often be violated in practice since the participating devices generally have different computational power and/or memory footprint. The varying infrastructure capabilities of the clients' devices imply that the models they are able to deploy differ in size, i.e., devices with more computational power and memory allow models of higher precision (e.g., 16-bit and 32-bit) while less capable devices may only handle low-precision models (e.g., those storing parameters in 2-bit or 4-bit precision). To satisfy local infrastructure constraints, not only should a model be represented and stored at reduced precision levels but the training process should also rely on operations at such levels (i.e., perform quantized training in low-bitwidth). Recently, (Yoon et al. 2022) studied the bitwidth heterogeneity problem in FL and proposed a progressive de-quantization of the received local models prior to their aggregation into a global model.

When the compute/storage resources are limited, on-device training of deep neural networks at full precision is rendered infeasible. To this end, recent works have explored model size reduction and the training that leads to low bitwidth weights/activations. These include binarized neural networks (BNN) and XNOR-Net which binarize the weights and activations of convolutional layers to reduce the computation complexity in the forward pass (Hubara et al. 2017; Rastegari et al. 2016). Specifically, in the forward pass the computationally expensive convolutional operations are implemented via bitwise operation kernels that evaluate the dot product of binary vectors $\mathbf{x}$ and $\mathbf{y}$

$$\mathbf{x} \cdot \mathbf{y} = \mathrm{bitcount}(\mathrm{and}(\mathbf{x}, \mathbf{y})), x_i, y_i \in \{0, 1\}\ \forall i,$$

where $\mathrm{bitcount}(\cdot)$ counts the number of bits in its argument. Such a kernel can be generalized to accommodate operations on M-bit fixed-point integer sequence x and K-bit fixed-point integer sequence y, incurring computation complexity of $\mathcal{O}(MK)$ (i.e., the complexity is proportional to the vector bitwidths). Related work (Zhou et al. 2016) presents DoReFa-Net which reduces the backpropagation computation cost by quantizing gradients in the backward pass, and studies the effect of the weight, activation and gradient bitwidth choice on the model performance. Subsequent works include techniques that deploy training without floating-point operations, gradient clipping, and training under mixed precision (Wen et al. 2017; Zhang et al. 2017; Zhu et al. 2020; Zhang et al. 2020). While all of the aforementioned methods investigate quantization in supervised learning settings, recently there has been interest in quantizing models for self-supervised learning. In (Cao et al. 2022), the authors propose self-supervised and quantization learning (SSQL), which contrasts features learned from the quantized and full precision models in order to provide reasonable accuracy of quantized models and boost the accuracy of the full precision model. While this work can balance the accuracy and bitwidth in resource constrained settings, it still requires full precision operations and gradient storage during training.

There have also been multiple efforts to ameliorate the impact of data heterogeneity on the performance of distributed learning systems (Karimireddy et al. 2020; Ghosh et al. 2020; Li et al. 2022). Recently, self-supervised learning (SSL) has been shown effective in distributed settings where the data is large-scale and imbalanced (Wang et al. 2022). In contrast to supervised learning, which requires large amount of labeled

data for model training, self-supervised learning defines a pre-train task that enables extracting expressive representations; those representations can then be utilized in various downstream tasks in computer vision and natural language processing (Chen et al. 2020; Chen and He 2021). While the majority of self-supervised learning methods focus on the centralized setting where the data is collected for central training, recent works have attempted to bridge the self-supervised and federated learning (He et al. 2021; Zhuang, Wen, and Zhang 2022; Makhija, Ho, and Ghosh 2022).

Preliminaries

Let us consider a federated learning system with n clients, where the clients rely on privately owned data to collaboratively learn a global model. Rather than communicating the data, the clients send to a coordinating central server the models trained locally. In turn, the server forms a global model by aggregating the received local models, and re-distributes the global model back to the clients for further local training. Let $D_k = \{x_{k,i}\}_{i=1}^{|D_k|}$ denote the data owned by client k, where $x_{k,i} \sim \mathcal{D}_k$ is a d-dimensional data point in D_k and $|D_k|$ denotes the number of local data points. The distribution $\mathcal{D}_k$ of data varies from one client to another, leading to system-wide data heterogeneity. The union $D = \bigcup_{k=1}^{n} D_k$ is assumed to be a dataset of uniform distribution, i.e., the data classes in D are balanced.

In self-supervised learning problems, an embedding function $f_{\mathbf{w}}(\cdot)$, parameterized by $\mathbf{w}$, is learned as a feature extractor for extracting expressive representations from data. In particular, we will denote the representation vector for point x by $f_{\mathbf{w}}(x)$; such a representation vector is then used for downstream tasks such as image classification or object detection. Contrastive learning is a popular methodology for identifying the embedding function $f_{\mathbf{w}}(\cdot)$ (Chen et al. 2020). The aim of contrastive learning is to find an embedding function such that the distance between the learned representations of data point x and its positive signal x^+ (generated from x) is small, while the distance between the representations of x and negative samples x^- (extracted from the training batch) is large.[1] A commonly used objective in contrastive learning is the InfoNCE loss; specifically, the local objective for client k is

$$\mathcal{L}_{CL,k}(\mathbf{w}) = \frac{1}{|D_k|} \sum_{i=1}^{|D_k|} -\log\left(\frac{\mathbb{D}_{k,i}^{+}}{\mathbb{D}_{k,i}^{+} + \sum_j \mathbb{D}_{k,i,j}^{-}}\right),$$

where $\mathbb{D}_{k,i}^{+} = \exp(-\mathbb{D}(f_{\mathbf{w}}(x_{k,i}), f_{\mathbf{w}}(x_{k,i}^{+}))/\tau$, $\mathbb{D}_{k,i,j}^{-} = \exp(-\mathbb{D}(f_{\mathbf{w}}(x_{k,i}), f_{\mathbf{w}}(x_{k,j}^{-}))/\tau$, and $\tau > 0$ is a temperature parameter controlling the sharpness of the exponential term. The function $\mathbb{D}(\cdot)$ is typically defined as the cosine distance, i.e., $\mathbb{D}(z_1, z_2) = -\frac{z_1 \cdot z_2}{\|z_1\|\|z_2\|}$. Note that instead of the negative sample term, an alternative SSL strategy adds to the objective a feature prediction function $g(\cdot)$ that helps avoid potential collapse i.e. low accuracy of the trained classifier layer (Chen and He 2021); in that case, the objective becomes

$$\mathcal{L}_{siam,k}(\mathbf{w}) = \frac{1}{|D_k|} \sum_{i=1}^{|D_k|} \mathbb{D}(g(f_{\mathbf{w}}(x_{k,i})), f_{\mathbf{w}}(x_{k,i}^{+})).$$

In federated learning, each client has a local data source D_k. The clients collaboratively search for $\mathbf{w}$ that minimizes the global objective function

$$\mathcal{L}(\mathbf{w}) = \sum_{k=1}^{n} \frac{|D_k|}{|D|} \mathcal{L}_k(\mathbf{w}). \tag{1}$$

To solve this distributed optimization without data sharing, numerous federated learning schemes have been proposed including the FedAvg algorithm (McMahan et al. 2017). At each iteration of FedAvg, central server samples a subset of clients which then run E local update epochs. The server collects updated models from the selected clients and aggregates them to form a global model, which is distributed to all clients for further local training/inference. The number of local training epochs, E, controls the computation-communication tradeoff, i.e., larger E leads to less frequent communication.

We further assume that participating clients have devices with varied compute, memory and communication capabilities, and therefore train and deploy models at different bitwidth precision. Let s_k denote the number of bits used to represent parameters $\mathbf{w}_k$ of the model deployed by client k. When training, client k utilizes bit convolutional kernels and conducts operations on s_k-bit objects rather than performing full precision (i.e., 32-bit floating-point) operations. When aggregating the collected local models into global model $\mathbf{w}_G$, the server takes into consideration that the models are trained at different precision levels. The server itself is assumed to be capable of performing operations at full precision. Finally, the global model is compressed to various low bitwidth representations and sent to the corresponding clients, i.e., client k receives updated model in s_k bit representation.

In our proposed framework, client k performs low-bit training to learn parameters $\mathbf{w}_k$ of its local model; note that the low-bit training and quantization are deployed in all steps, including the forward pass, backpropagation and model update. In each round of training, model parameters in s_k-bit representation are fed into the forward pass via bit convolution kernels, and the gradient computed from the backward propagation is quantized and stored. After updating the model by using gradient descent, the model parameters are quantized to s_k bits and stored.

Algorithm

The proposed federated quantized self-supervised learning scheme consists of training procedures deployed at clients' devices and the server, as detailed next.

At each client's side, a feature extraction model is trained in low bitwidth with layer-wise quantization in both forward and backward propagation pass. We denote the model parameters of client k in training epoch t by $\mathbf{w}_{k,t}^{(Q)}$; each parameter is

[1] Given a data sample x, its positive signal may be a variant obtained by adding noise, changing color, or clipping; negative signals include data samples from different classes, or simply other data samples in the same training batch.

represented using s_k bits. The codebook used for quantizing parameters of each layer of the model, $\mathbf{w}_{k,t}^{(Q)}$, has cardinality 2^{s_k}. The quantization centers (i.e., codebook entries) are typically determined via uniform quantization (Zhou et al. 2016),

$$r_o = \text{quantize}_{\mathbf{s}}(r_i) = \frac{1}{2^s - 1}\text{round}((2^s - 1)r_i),$$

which uses a uniform quantizer $\text{quantize}_{\mathbf{s}}$ to map a real-valued input $r_i \in [0, 1]$ to an s-bit output $r_o \in [0, 1]$. In general, parameters of deep neural networks do not follow uniform distribution and may vary in range. Therefore, the parameters may first need to be non-linearly transformed, e.g., by applying $\tanh$ function to limit their range to $[-1, 1]$. If $\tanh$ function is used, the quantized model weights are found as

$$r_o = 2\text{quantize}_s(\frac{\tanh r_i}{2\max(\tanh r_i)} + \frac{1}{2}) - 1,$$

where the maximum is taken over all parameters in the same layer. In the forward pass, we apply normalization to ensure that $\frac{\tanh r_i}{2\max(\tanh r_i)} + \frac{1}{2} \in [0, 1]$, where $\max(\tanh r_i)$ represents the maximum taken over all parameters in this layer. Following the quantization operation, output r_o is in $[-1, 1]$, which improves the stability of training.

In addition to quantizing model parameters, the proposed scheme quantizes activations as well. In particular, after the output of a layer passes through an activation function that guarantees the output to be within $[0, 1]$ range, the activation is quantized using $\text{quantize}_{\mathbf{s}}$.

To quantize gradients in backward propagation, we use layer-wise quantile quantization method. In particular, we rely on the inverse of the cumulative distribution function $F_{\mathbf{g}}$ of the gradients in a layer to define the quantile function $Quantile_{\mathbf{g}} = F_{\mathbf{g}}^{-1}$. The quantization centers are then specified by the end points of each histogram bin, $Quantile_{\mathbf{g}}(\frac{i}{2^s-1})$, for $i = 0, \ldots, 2^s - 1$. It has been observed in prior work on low-bitwidth training (Zhou et al. 2016) that to avoid significant performance degradation, the number of bits for gradient quantization should be no less than the number of bits for weight quantization (and may in fact need to be greater). In our algorithm, local gradient quantization uses 2 bits more than local weight quantization, e.g., if client k uses s_k bits for weight/activation quantization then it uses $s_k + 2$ bits for gradient quantization. Following the backward pass, the computed estimate of stochastic gradient, $\mathbf{g}_{k,t}$, is immediately quantized to $\mathbf{g}_{k,t}^{(Q)}$. Finally, updated weights are quantized according to $\mathbf{w}_{k,t+1}^{(Q)} = Q(\mathbf{w}_{k,t}^{(Q)} - \alpha_t \mathbf{g}_{k,t}^{(Q)})$.

In each communication round the server collects and de-quantizes local models, aggregates the de-quantized models, re-quantizes the resulting model to low bitwidth and finally distributes the models to clients. To elaborate, after the central server collects local low-bitwidth models, it uses a small buffer of unlabeled data D_g to de-quantize the collected local models to full precision. The de-quantization process differs from one local model to another and can be performed in parallel. In the scenario without bitwidth constraints, de-quantization can be viewed as a fine-tuning step on uniform unlabeled data; simulation results presented in the next sections demonstrate that such fine-tuning improves robustness in the full-precision model scenario.

After de-quantizing the collected models, the server forms their weighted average, $\mathbf{w}_{t,G} = \sum_{k=1}^{n} p_k \mathbf{w}_{k,t}^{\prime(Q)}$. The weights reflect performance discrepancy among local models caused by both local bitwidth and data heterogeneity, and are computed as

$$p_k = \frac{e^{-\mathcal{L}_{DQ_k}}}{\sum_{j=1}^{n} e^{-\mathcal{L}_{DQ_j}}}$$

where $\mathcal{L}_{DQ_k}$ represents the last epoch loss in the de-quantization process of model k. The server re-quantizes global model $\mathbf{w}_{t,G}$ to an s_k-bit version denoted by $\mathbf{w}_{k,t}^{(Q)}$ while running fine-tuning epochs on a uniformly distributed dataset stored in the global buffer D_g. The fine-tuning epochs use low-bit operations in order to guarantee that the fine-tuned model stays in s_k-bit representation. Note: both de-quantization and re-quantization use D_g for fine-tuning but de-quantization runs full precision operations while re-quantization runs low-bit operations.

Algorithm 1: Fed-QSSL: Federated quantized self-supervised learning under bitwidth and data heterogeneity

Initialization: Clients bitwidth configuration $\{s_k\}_{k=1}^n$, local dataset $\{D_k\}_{k=1}^n$, initial models $\{\mathbf{w}_{k,0}^{(Q)}\}_{k=1}^n$, global data buffer D_g, $t = 0$

Parameter: number of local training epoch E, step size $\{\alpha_t\}$

for each round $r = 1, 2,$ **do**
  {Local training at clients }
  for each client k in parallel **do**
    for $t = 0, 1, \cdots, E-1$ **do**
      Compute the compressed gradient $\mathbf{g}_{k,t+(r-1)E}^{(Q)}$
      Update the low-bit weights $\mathbf{w}_{k,t+1+(r-1)E}^{(Q)} = Q(\mathbf{w}_{k,t+(r-1)E}^{(Q)} - \alpha_{t+(r-1)E}\mathbf{g}_{k,t+(r-1)E}^{(Q)})$
      $t = t + 1$
    end for
  end for
  {Global operations at the server }
  The server collects local models $\{\mathbf{w}_{k,rE}^{(Q)}\}_{k=1}^n$
  The server de-quantizes $\{\mathbf{w}_{k,rE}^{(Q)}\}$ using D_g to obtain $\{\mathbf{w}_{k,rE}^{\prime(Q)}\}$ and loss $\{L_{DQ_k}\}$ for all $k \in [n]$
  The server aggregates $\mathbf{w}_{rE,G} = \sum_{k=1}^{n} p_k \mathbf{w}_{k,rE}^{\prime(Q)}$ using DQ-based losses
  The server re-quantizes the aggregation using D_g to s_k-bit model and distributes to client k for all $k \in [n]$
end for
Post-training Each client freezes the trained feature extractor and trains local linear probe (i.e., classification layer) for downstream tasks

After completing pre-training and learning the feature extractor model, each client freezes parameters of the feature extractor model and trains a low-bitwidth classifier on its local (labeled) data.

The described procedure, including both the steps executed by clients as well as those executed by the server, is formalized as Algorithm 1.

Remark. While Algorithm 1 assumes full client participation at each communication round, it is straightforward to incorporate random client sampling as used by the vanilla FedAvg algorithm in settings with a large number of clients. Moreover, when the number of clients is exceedingly large, it may be beneficial to cluster clients with similar data distributions and learn a model for each such cluster, ultimately allowing better utilization of the available resources.

Theoretical Analysis

In this section, we present analytical results that provide insights into Fed-QSSL, with a focus on the impact of low-bitwidth training on the convergence and robustness. For the sake of tractability, we consider the SSL formulation utilizing local objective defined as $\mathcal{L}_{SSL,k}(\mathbf{w}) = -\mathbb{E}[(\mathbf{w}(x_{k,i})+\xi_{k,i})^T(\mathbf{w}(x_{k,i})+\xi'_{k,i})]+\frac{1}{2}\|\mathbf{w}^T\mathbf{w}\|^2$, where $\xi_{k,i}$ and $\xi'_{k,i}$ denote random noise added to the data sample, while the global objective is defined as $\mathcal{L}_{SSL} = \sum_{k=1}^n \frac{|D_k|}{|D|}\mathcal{L}_{SSL,k}(\mathbf{w})$. This objective is obtained from the InfoNCE loss $\mathcal{L}_{CL,k}$ by replacing normalization via negative signals by an alternative regularization term (Wang et al. 2022). Optimizing $\mathcal{L}_{SSL}$ is equivalent to minimizing $\mathcal{L}(\mathbf{w}) = \|\bar{X} - \mathbf{w}^T\mathbf{w}\|^2$ where $\bar{X} = \sum_k \frac{|D_k|}{|D|}X_k$ and $X_k = \mathbb{E}_{x\sim D_k}(xx^T) = \frac{1}{|D_k|}\sum_{i=1}^{|D_k|} x_{k,i}x_{k,i}^T$, the empirical covariance matrix of client k's data. Since the aim of this analysis is to assess the impact of low-bitwidth operations at client devices on the convergence and robustness of federated learning, we simplify operations at the server by replacing the sophisticated de-quantizer in Algorithm 1 by a layer-wise codebook mapping low-bitwidth weights to floating-point values according to $\mathbf{w}'^{(Q)}_{k,t} = \mathbf{w}^{(Q)}_{k,t}$. The subsequent aggregation follows FedAvg and computes $\mathbf{w}_{t,G} = \sum_{k=1}^n \frac{|D_k|}{|D|}\mathbf{w}'^{(Q)}_{k,t}$, while re-quantization uses a $\tanh$ based quantizer.

For simplicity, we assume that clients quantize weights/activations using the same bitwidth, i.e., $s_1 = s_2 = \cdots = s_n = R$; moreover, local gradient quantization is performed using the same number of bits. When bitwidth varies across client devices, the devices with higher bitwidth typically have smaller quantization error. Our forthcoming analysis can be viewed as providing error bounds for a system in which clients deploy bitwidth no less than R. Local datasets are generated in a distributed and $2n$-way manner, i.e., local dataset at client k is generated such that the labels of data samples are skewed to classes $2k-1$ and $2k$, with a few samples coming from other classes. This distribution is close to the pathologically non-iid case with two dominant classes. More details about heterogeneous data generation are provided in Appendix.

We consider three types of quantization errors: $\epsilon_{\mathbf{w}}$ denotes the model parameter quantization error, ϵ_g denotes the gradient quantization error, and ϵ_r denotes the quantization error induced by the server when re-quantizing the aggregated model. Below we use subscripts k and t to indicate the client and the epoch, respectively. Quantization can be viewed as adding noise to the full precision values, i.e., $\mathbf{g}^{(Q)}_{k,t} = \mathbf{g}_{k,t} + \epsilon_{g_k,t}$. In quantized training at client k, the update can be expressed as

$$\mathbf{w}^{(Q)}_{k,t+1} = \mathbf{w}^{(Q)}_{k,t} - \alpha_t(\mathbf{g}_{k,t} + \epsilon_{g_k,t}) + \epsilon_{\mathbf{w}_k,t}$$

where $\mathbf{w}^{(Q)}_{k,t}$ denotes the low-bitwidth model parameters at time t, $\epsilon_{g_k,t}$ is the noise added to the gradient in the quantization step, and $\epsilon_{\mathbf{w}_k,t}$ denotes the noise added to the model parameters after gradient update as the parameters retain low-bitwidth representation. After E local training epochs, the server collects the models and aggregates them according to

$$\mathbf{w}_{t+E,G} = \sum_{k=1}^n \frac{|D_k|}{|D|}\mathbf{w}^{(Q)}_{t+E,k} = \sum_{k=1}^n \frac{|D_k|}{|D|}\mathbf{w}^{Q}_{k,t} - \sum_{s=0}^{E-1}[\alpha_{t+s}(\mathbf{g}_{k,t+s} + \epsilon_{g_k,t+s}) - \epsilon_{\mathbf{w}_k,t+s}].$$

Since the aggregated model is not necessarily in low bitwidth, an additional re-quantization step is required to form its quantized version $\mathbf{w}^{(Q)}_{t+E,G} = \mathbf{w}_{t+E,G} + \epsilon_{r,t+E}$, which is then sent to the clients.

Quantization centers used for layer-wise scalar quantization of the model parameters and gradients are found via companding quantization. Specifically, full-precision input $\mathbf{x}$ is transformed by a nonlinear function c, e.g., $\tanh$ function, and then uniformly quantized. The output $Q(\mathbf{x})$ is generated by taking the value of the inverse function, c^{-1}, of the quantized value (Sun and Goyal 2011). Uniform quantization is a special case of companding quantization obtained by setting $c(\mathbf{x}) = \mathbf{x}$. In Fed-QSSL, local quantizers can also be viewed as special cases of the companding quantizers.

To proceed with the analysis, we make the following assumption on the quantizers.

Assumption 1. *All quantization operators in the low-bit training are unbiased.*

This assumption is commonly encountered in prior work (see, e.g., (Reisizadeh et al. 2020)); an example is the stochastic quantizer, i.e., given a sequence of quantization centers $Q_1 \le \cdots \le Q_{2^R}$, for $x \in [Q_j, Q_{j+1}]$ we have $Q(x) = Q_j$ with probability $\frac{Q_{j+1}-x}{Q_{j+1}-Q_j}$ and $Q(j) = Q_{j+1}$ with probability $\frac{x-Q_j}{Q_{j+1}-Q_j}$. This assumption implies that quantization errors $\epsilon_{\mathbf{w}}$, ϵ_g, and ϵ_r are zero-mean.

We further make the following assumption on gradient estimates, frequently encountered in literature.

Assumption 2. *Expected gradient estimate is unbiased and bounded,* $\mathbb{E}_t\|\mathbf{g}_{k,t}\|^2 \le G^2$.

The following lemma provides a bound on the quantization errors (for proof please see Appendix).

Lemma 1. *Suppose Assumptions 1-2 hold, and that client k computes update of the quantized model parameters $\mathbf{w}_{k,t+1}^{(Q)}$ at bitwidth $s_k = R$. Then the gradient quantization error $\epsilon_{g_k,t}$ satisfies $\mathbb{E}_t[\|\epsilon_{g_k,t}\|^2] = \mathcal{O}(G^2/2^{2R})$, and the local re-quantization error $\epsilon_{\mathbf{w}_k,t}$ satisfies $\mathbb{E}_t[\|\epsilon_{\mathbf{w}_k,t}\|^2] = \mathcal{O}(\alpha_t G^2/2^{2R})$.*

This lemma indicates that by viewing the gradient update $\alpha_t(\mathbf{g}_{k,t} + \epsilon_{g_k,t})$ as a perturbation of $\mathbf{w}_{k,t}^{(Q)}$, the variance of the weight quantization error $\epsilon_{\mathbf{w}_k,t}$ in the local update is of the order $\mathcal{O}(\alpha_t G^2)$. It further implies that the variance of the re-quantization error $\epsilon_{r,t}$ in the global update is of the order $\mathcal{O}(\alpha_t G^2 E)$ when $\alpha_{t-E+1} = \cdots = \alpha_t$.

The next corollary readily follows from Lemma 1.

Corollary 1. *Instate the assumptions of Lemma 1, and let all clients use the same learning rate $\alpha_{t-E+1} = \cdots = \alpha_t$. Then there exists $G_q = \mathcal{O}(G^2/2^{2R})$ such that $\mathbb{E}_t[\|\epsilon_{\mathbf{w}_k,t}\|^2] \leq \alpha_t G_q^2$ and $\mathbb{E}_t[\|\epsilon_{r,t}\|^2] \leq \alpha_t G_q^2$.*

Next, we consider the convergence of the quantized SSL training. Note that objective $\mathcal{L} = \|\bar{X} - \mathbf{w}^T\mathbf{w}\|^2$ is generally a non-convex and non-smooth function. In our analysis, we consider a class of functions that satisfies the ρ-weak convexity.

Definition 1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is ρ-weakly convex if $f(x) + \frac{\rho}{2}\|x\|^2$ is convex.*

Generally, any function of the form $f(x) = g(h(x))$, where $g(\cdot)$ is convex and Lipschitz and $h(\cdot)$ is a smooth map with Lipschitz Jacobian, is weakly-convex. Clearly, the federated SSL objective satisfies ρ-weak convexity with $\rho \geq 4\|\bar{X}\|$. To analyze weakly-convex objective function $\mathcal{L}$, we introduce the Moreau envelope

$$\phi_\lambda(x) := \min_y \{\mathcal{L}(y) + \frac{1}{2\lambda}\|y - x\|^2\}$$

and define the corresponding proximal map as

$$\text{prox}_{\phi_\lambda(x)} := \text{argmin}_y \{\mathcal{L}(y) + \frac{1}{2\lambda}\|y - x\|^2\}.$$

We use $\|\nabla\phi_\lambda(x)\|$ as the convergence indicator. Intuitively, small norm of the gradient of $\phi_\lambda(x)$ implies that x is close to a point $\hat{x}$ that is stationary for ϕ. For Fed-QSSL, the following result holds.

Theorem 1. *Suppose all assumptions of Lemma 1 and Corollary 1 hold. For all $\bar{\rho} > \rho$, after T communication rounds of Fed-QSSL*

$$\frac{1}{\sum_{t=0}^T \alpha_{tE}} \sum_{t=0}^T \alpha_{tE}\mathbb{E}[\|\nabla\phi_{\frac{1}{\bar{\rho}}}(\mathbf{w}_{tE,G}^{(Q)})\|^2] \leq \frac{E\bar{\rho}}{\bar{\rho} - \rho} \times \frac{\phi_{\frac{1}{\bar{\rho}}}(\mathbf{w}_{0,G}^{(Q)}) - \min\phi + \bar{\rho}(G^2\sum_{t=0}^T \alpha_{tE}^2 + 3G_q^2\sum_{t=0}^T \alpha_{tE})}{\bar{\rho}\sum_{t=0}^T \alpha_{tE}},$$

where E denotes the number of local training epochs per communication round (for a total of tE training epochs after t communication rounds) and $\mathbf{w}_{0,G}^{(Q)}$ is the quantized parameter initialization.

Theorem 1 implies convergence of the algorithm to a nearly stationary state of the objective function. When α_t decreases as $\mathcal{O}(1/\sqrt{t})$, the upper bound vanishes as $t \to \infty$. The optimal solution $\mathbf{w}^*$ of the global objective satisfies $\|\nabla\phi_{\frac{1}{\bar{\rho}}}(\mathbf{w}^*)\|^2 = 0$; the vanishing upper bound implies proximity to a neighborhood of the optimal solution.[2] As the quantization rate R approaches full precision rate, G_q vanishes and the upper bound collapses to the value obtained by vanilla stochastic gradient descent analysis.

It is further of interest to analyze robustness of the learned representations. To this end, we first formally define the representation vector.

Definition 2. *(Wang et al. 2022) Let $\mathcal{S} \subset \mathbb{R}^d$ be the subspace spanned by the rows of the learned feature matrix $\mathbf{w} \in \mathbb{R}^{m\times d}$ for the embedding function $f_\mathbf{w}(x) = \mathbf{w}x$. The representability of $\mathcal{S}$ is defined as the vector $r = [r_1, \cdots, r_d]^T$ such that $r_i = \|\Pi_\mathcal{S}(e_i)\|^2$ for $i \in [d]$, where $\Pi_\mathcal{S}(e_i)$ denotes the projection of standard basis e_i onto $\mathcal{S}$ and thus $r_i = \sum_{j=1}^s \langle e_i, v_j\rangle^2$, $s = \dim(\mathcal{S})$ and $\{v_j\}$ is the set of orthonormal bases of $\mathcal{S}$.*

Vector r introduced in this definition quantifies representability by comparing unit bases of different feature spaces. A good feature space should be such that the entries of r are large, indicating its standard unit bases can well represent $e_1, \ldots, e_d$, the unit bases in $\mathbb{R}^d$.

Theorem 2. *Suppose assumptions of Theorem 1 hold and $n = \Theta(d^{1/20})$. In a $2n$-way classification task, when in local training the update is ϵ away from the optimal solution $\mathbf{w}_k^*$, the representation vector learned by client k with high probability satisfies $\frac{d^{2/5} - \mathcal{O}(d^{-2/5}) + 2e_j^T(\mathbf{w}_k^*)^T\epsilon e_j + (e_j^T\epsilon)^2}{d^{2/5} + \mathcal{O}(d^{-2/5}) + \|2(\mathbf{w}_k^*)^T\epsilon + \epsilon^T\epsilon\|} \leq r_i^k \leq 1$ for $i \in [n]\backslash\{k\}$. As for the global objective, when the update is ϵ_1 away from the optimal solution $\mathbf{w}^*$, the learned representation vector $\bar{r}$ with high probability satisfies $\frac{d^{2/5} - \Theta(d^{7/20}) + \mathcal{O}(d^{-1/20}) - \mathcal{O}(d^{2/5}) + 2e_j^T(\mathbf{w}^*)^T\epsilon_1 e_j + (e_j^T\epsilon_1)^2}{d^{2/5} - \Theta(d^{7/20}) + \mathcal{O}(d^{-1/20}) + \|2(\mathbf{w}^*)^T\epsilon_1 + \epsilon_1^T\epsilon_1\|} \leq \bar{r}_i \leq 1$ for all $i \in [n]$.*

Theorem 2 implies that as ϵ_1 vanishes, the entries in representation vectors have strictly positive lower bound and thus do not vanish. This theorem states that the representation space learned from the SSL objectives is such that for the $2n$ basis directions that generate the data, any two clients have similar representability. The theorem further implies that the learned representation vectors are not biased towards local data distributions and are capable of performing well on uniformly distributed data, indicating robustness of the scheme. For the proof of Theorem 2, please see Appendix.

Experimental Results

We simulate an FL system with 10 clients, where the participating clients have different bitwidth configuration and local data distributions. We deploy two bitwidth configurations: (1) the configuration with gradually increasing bitwidth: 20%4-bit, 30%6–bit, 30%8-bit and 20%12-bit models; and (2) the

[2]While using gradient descent estimates with bounded variance typically leads to converge to local minima (Mertikopoulos et al. 2020), a property of the SSL objective is that all local minima are global minima (Jin et al. 2017).

Algorithm	Global Acc	Local Acc
FedAvg	17.70	64.98
FedProx	14.51	68.46
FedPAQ	16.68	62.96
Fed-SimCLR	39.09	77.30
Fed-SimSiam	36.73	76.87
Fed-QSSL	**59.31**	**82.50**
Fed-SimCLR (Full)	40.76	80.56
Fed-QSSL (Full)	**72.26**	**90.01**

Table 1: Experiments on CIFAR-10 with model bitwidth configuration 20%4-bit, 30%6–bit, 30%8-bit and 20%12-bit.

Algorithm	Global Acc	Local Acc
FedAvg	20.77	66.64
FedProx	18.70	75.60
FedPAQ	18.56	76.57
Fed-SimCLR	39.12	77.43
Fed-SimSiam	36.44	75.21
Fed-QSSL	**62.26**	**83.03**
Fed-SimCLR (Full)	40.76	80.56
Fed-QSSL (Full)	**72.26**	**90.01**

Table 2: Experiments on CIFAR-10 with model bitwidth configuration 50%6-bit and 50%12-bit.

configuration with a skewed bitwidth: 50%6-bit and 50%12-bit models. The experiments involve CIFAR-10 and CIFAR-100 datasets, both for image classification tasks (Krizhevsky, Hinton et al. 2009). The datasets are distributed to clients in a non-iid fashion according to Dirichlet(0.1) distribution (Bibikar et al. 2022). Simulations were executed on AMD Vega 20 GPUs. The learning rates were set as in the following prior work: for the supervised learning schemes using low-bit operations we deploy learning rates as in (Zhou et al. 2016), while for self-supervised learning schemes we follow (Wang et al. 2022). In all simulations, the deployed neural networks were based on ResNet-18 architecture (He et al. 2016). The results are reported after 100 communication rounds, with clients running $E = 20$ local epochs between any two rounds in CIFAR-10 simulations and $E = 5$ in CIFAR-100 simulations. In the implementation of Fed-QSSL, we rely on the SimCLR method (Chen et al. 2020) for the self-supervision part of the algorithm.[3]

We compare Fed-QSSL with two groups of algorithms: federated supervised learning (SL) and self-supervised algorithms. The SL algorithms include FedAvg (McMahan et al. 2017), classic FL technique performing simple averaging of local models at each communication round; FedProx (Li et al. 2020), an FL scheme addressing client data heterogeneity by adding an ℓ_2-norm regularizer to local objectives to prevent divergence of local updates from the global model; and FedPAQ (Reisizadeh et al. 2020), a communication-efficient FL algorithm where clients transmit quantized updates to reduce

[3]The codes are available at https://github.com/YiyueC/Fed-QSSL.

Algorithm	Global Acc	Local Acc
Fed-SimCLR	14.79	23.46
Fed-SimSiam	12.24	13.59
Fed-QSSL	**21.44**	**30.91**
Fed-SimCLR (Full)	15.18	30.26
Fed-QSSL (Full)	**28.75**	**43.56**

Table 3: Experiments on CIFAR-100 with bitwidth configuration 50%6-bit and 50%12-bit.

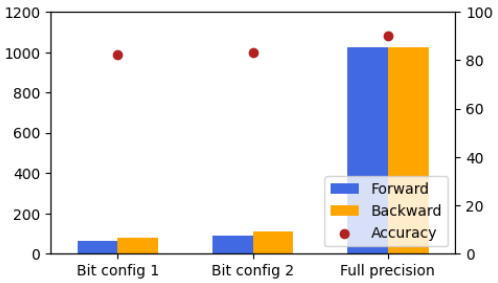


Figure 1: The left y-axis indicates the number of bitwise operations while the right y-axis indicates the local accuracy achieved by Fed-QSSL.

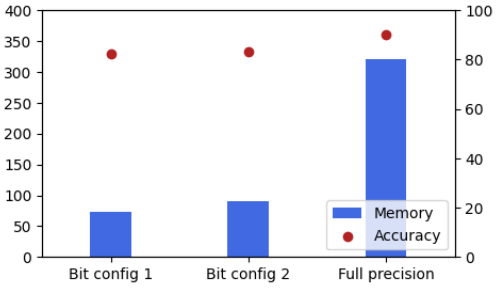
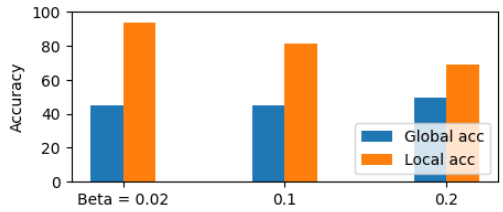


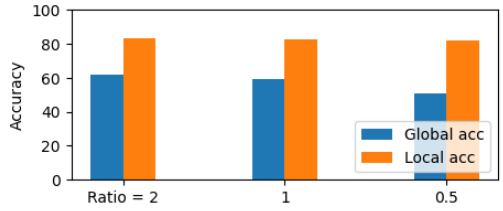
Figure 2: The left y-axis indicates the memory consumption while the right y-axis indicates the local accuracy achieved by Fed-QSSL.

uplink communication cost. As for the SSL algorithms, we consider Fed-SimCLR (Chen et al. 2020; Wang et al. 2022) which uses contrastive learning objective to learn a global feature extraction model; and Fed-SimSiam (Chen and He 2021) which considers only positive pairs of data points and learn meaningful features by leveraging a feature predictor function and stop-gradient operation. While the baseline algorithms are not originally meant to support local low-bit training, we apply low-bitwidth operations to satisfy resource constraints. In each table of results, the last two rows (labeled as "Full") correspond to full precision (32-bit) models, i.e., models with no bitwidth constraints.

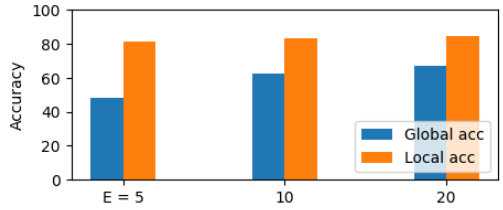
Fed-QSSL is compared to the baselines in terms of global accuracy (Global Acc), reflective of robustness, and local accuracy (Local Acc), reflecting personalization. SSL methods train a universal linear classifier on top of the frozen global feature extraction model and evaluate the classification ac-



(a) Dirichlet distribution parameter β.



(b) The ratio of the global buffer size and the local dataset size.



(c) The number of local training epochs E.

Figure 3: Ablation study.

curacy on a uniformly distributed test dataset. Training of a linear classifier to be used for testing global model accuracy is conducted at full precision on a centralized training dataset as in (Wang et al. 2022); this is done only to validate robustness of SSL methods, in real applications no such universal linear classifier is required. SSL methods use the aggregated model and test on uniform testing dataset. To evaluate local accuracy (i.e., the accuracy on local, generally non-IID datasets), SSL methods train a local linear classifier on top of the frozen low-bitwidth feature extractor while meeting client's bitwidth constraints and run it on a local non-iid testing dataset, while SL methods use a local low-bitwidth classification model and compute the accuracy on local testing data.

Results

The results on CIFAR-10 under the first bitwidth configuration are reported in Table 1. Overall, the global accuracy of SSL schemes is higher than SL methods, implying that SSL algorithms tend to learn more accurate representations in data heterogeneous FL scenario. As for the personalization, SSL algorithms achieve higher accuracy on local data, suggesting they learn meaningful representations that are then used in the downstream classification task. It is worth noting that SSL algorithms with higher global accuracy also achieve higher local accuracy; this is because the learned robust feature extractor extracts expressive representations from heterogeneous data which help in downstream classification tasks. Among SL algorithms, however, there exists a trade-off between robustness and personalization, where the performance of models with high global accuracy may suffer locally. Specifically, Fed-QSSL performs better in both bitwidth-heterogeneous and no-constrained (full precision) scenarios due to the use of server-side operations on the global buffer; operations on server facilitate learning robust representations that further improve local performance. Table 2 reports results on CIFAR-10 under the second bitwidth configuration. There, SSL algorithms again demonstrate more robust and personalized performance in face of bitwidth and data heterogeneity. As the permitted bitwidth grows, the algorithms reach higher accuracy. Nevertheless, under the bitwidth constraints Fed-QSSL still achieves the highest accuracy. Finally, Table 3 reports results on the more challenging CIFAR-100 dataset. As can be seen there, Fed-QSSL perform better than other SSL benchmarks under both low-bitwidth as well as full precision scenarios.

We next compare the computational cost and memory requirements of Fed-QSSL to those of the full precision scheme. In particular, for the two bitwidth configurations and the full precision scenario considered above, we report the number of bitwise operations and memory footprint in the local training (see Figure 1 and Figure 2). In Figure 1, we show the number of bitwise operations for the considered bitwidth configurations; as can be seen there, in both forward and backward pass the computational cost of Fed-QSSL is much smaller than that of the full precision scheme. In low-bit configurations, the backward propagation is consuming more computation because the gradient are represented using 2 bits more than the weights. Figure 2 shows that Fed-QSSL achieves significant memory savings while still providing competitive accuracy.

Lastly, Figures 3(a)-3(c) present results of ablation studies. When increasing Dirichlet distribution parameter β, local data distributions become more uniform and the global accuracy increases. The local accuracy, however, decreases since the clients start facing more classes and thus need to execute more challenging classification tasks (Fig. 3(a)). When smaller size global buffer is used and the ratio of the size of D_g and the size of local data decreases, the global accuracy deteriorates while the local accuracy remains mostly unchanged (Fig. 3(b)). Finally, as more training epochs are used in local training, Fed-QSSL achieves higher accuracy (Fig. 3(c)).

Conclusion and Future Work

We introduced the federated quantized self-supervised learning (Fed-QSSL) algorithm, an effective framework for FL in bitwidth and data heterogeneity settings. We analytically studied the impact of low-bit training on the convergence and robustness of FL, and experimentally demonstrated that Fed-QSSL achieves more robust and personalized performance than benchmarking algorithms. Future work may include an extension to large-scale settings where it is meaningful to cluster clients with similar data distributions, and train per-cluster models. For such systems, it is of interest to develop schemes that aim to optimally manage utilization of the available resources.

References

Bibikar, S.; Vikalo, H.; Wang, Z.; and Chen, X. 2022. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6080–6088.

Cao, Y.-H.; Sun, P.; Huang, Y.; Wu, J.; and Zhou, S. 2022. Synergistic self-supervised and quantization learning. In *European Conference on Computer Vision*, 587–604. Springer.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, X.; and He, K. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 15750–15758.

Chen, Y.; Hashemi, A.; and Vikalo, H. 2021a. Communication-efficient variance-reduced decentralized stochastic optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 67(12): 6583–6594.

Chen, Y.; Hashemi, A.; and Vikalo, H. 2021b. Decentralized optimization on time-varying directed graphs under communication constraints. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3670–3674. IEEE.

Chen, Y.; Sun, X.; and Jin, Y. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE transactions on neural networks and learning systems*, 31(10): 4229–4238.

Diao, E.; Ding, J.; and Tarokh, V. 2020. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33: 19586–19597.

He, C.; Yang, Z.; Mushtaq, E.; Lee, S.; Soltanolkotabi, M.; and Avestimehr, S. 2021. Ssfl: Tackling label deficiency in federated learning via personalized self-supervision. *arXiv preprint arXiv:2110.02470*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1): 6869–6898.

Jiang, D.; Shan, C.; and Zhang, Z. 2020. Federated learning algorithm based on knowledge distillation. In *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 163–167. IEEE.

Jiang, Y.; Konevcnỳ, J.; Rush, K.; and Kannan, S. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*.

Jin, C.; Ge, R.; Netrapalli, P.; Kakade, S. M.; and Jordan, M. I. 2017. How to escape saddle points efficiently. In *International conference on machine learning*, 1724–1732. PMLR.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, Q.; Diao, Y.; Chen, Q.; and He, B. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 965–978. IEEE.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33: 2351–2363.

Makhija, D.; Ho, N.; and Ghosh, J. 2022. Federated self-supervised learning for heterogeneous clients. *arXiv preprint arXiv:2205.12493*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, 1273–1282.

Mertikopoulos, P.; Hallak, N.; Kavis, A.; and Cevher, V. 2020. On the almost sure convergence of stochastic gradient descent in non-convex problems. *Advances in Neural Information Processing Systems*, 33: 1117–1128.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, 525–542. Springer.

Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; and Pedarsani, R. 2020. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, 2021–2031. PMLR.

Servetnyk, M.; Fung, C. C.; and Han, Z. 2020. Unsupervised federated learning for unbalanced data. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 1–6. IEEE.

Sun, J. Z.; and Goyal, V. K. 2011. Scalar quantization for relative error. In *2011 Data Compression Conference*, 293–302. IEEE.

Wang, L.; Zhang, K.; Li, Y.; Tian, Y.; and Tedrake, R. 2022. Does Learning from Decentralized Non-IID Unlabeled Data Benefit from Self Supervision? In *The Eleventh International Conference on Learning Representations*.

Wen, W.; Xu, C.; Yan, F.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30.

Yoon, J.; Park, G.; Jeong, W.; and Hwang, S. J. 2022. Bitwidth heterogeneous federated learning with progressive weight dequantization. In *International Conference on Machine Learning*, 25552–25565. PMLR.

Zhang, H.; Li, J.; Kara, K.; Alistarh, D.; Liu, J.; and Zhang, C. 2017. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In *International Conference on Machine Learning*, 4035–4043. PMLR.

Zhang, X.; Liu, S.; Zhang, R.; Liu, C.; Huang, D.; Zhou, S.; Guo, J.; Guo, Q.; Du, Z.; Zhi, T.; et al. 2020. Fixed-point back-propagation training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2330–2338.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

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

Zhu, F.; Gong, R.; Yu, F.; Liu, X.; Wang, Y.; Li, Z.; Yang, X.; and Yan, J. 2020. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1969–1979.

Zhuang, W.; Wen, Y.; and Zhang, S. 2022. Divergence-aware federated self-supervised learning. *arXiv preprint arXiv:2204.04385*.