Mixed-Precision Quantization for Federated Learning on Resource-Constrained Heterogeneous Devices

Huancheng Chen
University of Texas at Austin
huanchengch@utexas.edu

Haris Vikalo
University of Texas at Austin
hvikalo@ece.utexas.edu

Abstract

While federated learning (FL) systems often utilize quantization to battle communication and computational bottlenecks, they have heretofore been limited to deploying fixed-precision quantization schemes. Meanwhile, the concept of mixed-precision quantization (MPQ), where different layers of a deep learning model are assigned varying bit-width, remains unexplored in the FL settings. We present a novel FL algorithm, FedMPQ, which introduces mixed-precision quantization to resource-heterogeneous FL systems. Specifically, local models, quantized so as to satisfy bit-width constraint, are trained by optimizing an objective function that includes a regularization term which promotes reduction of precision in some of the layers without significant performance degradation. The server collects local model updates, de-quantizes them into full-precision models, and then aggregates them into a global model. To initialize the next round of local training, the server relies on the information learned in the previous training round to customize bit-width assignments of the models delivered to different clients. In extensive benchmarking experiments on several model architectures and different datasets in both iid and non-iid settings, FedMPQ outperformed the baseline FL schemes that utilize fixed-precision quantization while incurring only a minor computational overhead on the participating devices.

1. Introduction

Federated Learning (FL) [27, 32, 37, 52] paradigm enables collaboration among numerous distributed devices (clients) while protecting their privacy by avoiding collection of data stored at those devices. The vanilla FL algorithm, FedAvg [37], deploys a client-server framework in which the server periodically collects locally trained/updated models from the clients and aggregates them into the global model. A number of of follow-up studies [44, 54, 55, 62] explored convergence guarantees for FedAvg with convex and non-convex loss functions in the settings where clients' data is independent and identically distributed (i.i.d.). However, in real-world scenarios, the participating clients' data is likely to be non-i.i.d., which has detrimental effects on the convergence properties of FedAvg [33]. This motivated a considerable amount of research aiming to address the challenges of statistical heterogeneity in federated learning [2, 5–7, 27, 30–32, 47, 48, 52, 65]. On a related note, clients in large-scale FL systems are likely to have varying communication and computational resources at their disposal, requiring development of more sophisticated, constraints-aware schemes [26]. Recently, there has been a growing interest in compressing models so that they can be deployed on devices having limited resources [10]. A particularly effective model compression technique is that of model quantization [20, 38, 39, 42, 43, 61].

The use of model quantization in FL has primarily been motivated by the high communications costs that arise when the server collects updates from a potentially very large number of clients. Reducing the precision of local updates by using lower bit-width to represent model parameters allows the clients with limited bandwidth to efficiently communicate with the server [17, 20, 24, 39, 40, 42, 43]. However, existing quantization-based FL algorithms expect each client to learn a full-precision local model regardless of the potential constraints on the client's computational resources. In real-world settings, clients such as smart phones or wearable devices may not have sufficient amount of memory to allow full-precision model training. Nevertheless, there are only few prior studies considering quantization-aware training in FL [1, 13, 25, 38, 61], all of them implementing *fixed-precision quantization* (FPQ) of local models. The recent advancements in *mixed-precision quantization* (MPQ) [1, 11, 51, 53, 56, 58, 59] remain unexplored in the federated learning settings. Adapting the existing MPQ schemes to federated learning is far from straightforward. The existing MPQ schemes typically train a full-precision model to enable computing the quantization errors which are then used to optimize layer-wise bit-width allocation. Both search-based [53, 56] and optimization-based [51, 59, 64] MPQ methods consume significantly

more computation than the FL training itself.

The aim of this paper is to introduce mixed-precision quantization to federated learning, developing an efficient framework that addresses the challenge of resource heterogeneity in FL. In particular, we study federated learning system where the clients deploy local models at average bit-width that may vary from one device to another. In such scenarios, clients with low average bit-width budgets cannot run computationally intensive MPQ methods. We propose FedMPQ, a novel **Fed**erated learning algorithm with **M**ixed-**P**recision **Q**uantization, which enables training of quantized local models within the allocated bit-width budget. FedMPQ first initializes local models as fixed-precision quantized networks that satisfy clients' average bit-width budget, and then converts these quantized networks into a representation that allows bit-level sparsity-promoting training. In particular, while learning local models whose parameters admit binary representation, the clients deploy a group Lasso regularization term which imposes a trade-off between the task loss and bit-sparsity. The precision of layers that end up having parameters which exhibit higher degree of sparsity is reduced to allow increasing precision of other layers. During the aggregation step, FedMPQ employs the *pruning-growing* strategy where the server aggregates clients' models (locally trained at potentially different bit-widths), resulting in the global model. Before transmitting the global model to a client, the bit-width of the model is adjusted to match the client's bit-width budget. To evaluate the effectiveness of FedMPQ, we conducted experiments on CIFAR10, CIFAR100 and Tiny-Imagenet [29]. FedMPQ outperforms the baseline in non-i.i.d. data settings and achieves performance similar to the FPQ8 baseline even though a subset of the clients train their models at a very low precision. The contributions of the paper are summarized as follows:

- We propose a method for mixed-precision quantization in FL which does not require training full-precision models on devices.
- We introduce a *pruning-growing* strategy for allocating layer-wise bit-width without a need for computationally expensive procedures that may violate resource constraints.
- We conduct extensive experiments in non-i.i.d. FL settings where the clients have heterogeneous computational resources, demonstrating the performance of the proposed FedMPQ.

2. Related Work

2.1. Quantization for Federated Learning

Much of the existing research on federated learning under resource constraints focuses on quantizing local updates for the purpose of reducing communication bandwidth [20, 61]. The milestone work, FedPAQ [39], presents a federated learning framework where each client communicates quantized local updates to the server, and provides analytical convergence guarantees for both strongly-convex and non-convex settings. FedCOMGATE [16] extends the ideas of FedPAQ to introduce a local gradient tracking scheme mitigating detrimental effect introduced when learning from non-i.i.d. data. UVeQFed [43] takes a step further, utilizing vector quantization to move from lossy to lossless compression. The follow-up studies [20, 24, 36, 42] propose adaptive quantization for local updates in communication-constrained settings while still requiring clients to locally train full-precision models.

AQFL [1] took a step towards mitigating the detrimental effects of computational heterogeneity by training quantized models with bit-widths proportional to the clients' computational resources. The follow-up works [8, 13, 61] developed a series of methods improving the server's aggregation of local updates quantized at varying levels. However, these methods are limited to fixed-precision quantization, assigning the same bit-width to the entire model. This motivates us to explore mixed-precision quantization in FL schemes as an alternative approach to learning in computationally heterogeneous scenarios.

2.2. Mixed-Precision Quantization

Aiming to enable stronger expressiveness of learned models, mixed-precision quantization (MPQ) assigns different bit-widths to different layers/modules of the models. The MPQ strategies, which generally attempt to assign bit-widths in proportion to the importance of different layers, can be organized in three categories: (1) search-based, (2) optimization-based, and (3) metric-based.

Search-based methods such as HAQ [53] and AutoQ [34] utilizes reinforcement learning (RL) [46] to pursue optimal bit-widths where the model performance is set as the reward. DNAS [56] and SPOS[14] apply neural architecture search (NAS) to explore the quantization space which is growing exponentially with the number of layers. Since both RL-based and NAS-based methods require an enormous amount of computation, it is unrealistic to implement them in FL settings.

Optimization-based methods approach MPQ from an optimization perspective by formulating the bit-width allocation problem using differentiable variables and applying the straight-through [3, 51, 59, 64] or gumbel-softmax [4, 15, 21, 23] estimator. However, this leads to mixed-integer programming problems which are NP-hard, rendering the use of optimization-based methods in FL scenarios practically infeasible.

In contrast to the search-based and optimization-based methods, metric-based methods leverage a variety of metrics to evaluate the importance of layers and subsequently

decide on the bit-width allocation. Such metrics include the eigenvalues of the Hessian matrix [9, 11, 12, 60], orthogonality conditions [35], entropy measures [45], synaptic flow [49] and learnable layer-wise importance [50]. The computation of the aforementioned metrics is relatively expensive as it requires the information from full-precision models. Recently, BSQ [58] presented a bit-pruning MPQ strategy that achieves high compression rate while preserving model performance; however, BSQ simulates binary representation with floating-point values, making it impractical for FL settings.

3. Methodology

3.1. Federated Learning with Quantization

In a cross-device scenario with N clients, where client n owns a private dataset $\mathcal{D}_n$, the standard federated learning (FL) considers training a single global model $\mathbf{W}$ by minimizing the loss (empirical risk)

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) = \sum_{n=1}^{N} p_n \mathcal{L}_n(\mathbf{W}), \tag{1}$$

where $\mathcal{L}_n(\cdot)$ is the local loss function on $\mathcal{D}_n$ and $p_n \in [0,1]$ denotes the weight assigned to client n. At round t of FedAvg [37], a widely used FL algorithm, each participating client locally trains a model $\mathbf{W}_n^t$ on local data and communicates it to the server; the server aggregates the collected local models to form the global model $\mathbf{W}^t = \sum_{k=1}^{N} p_n \mathbf{W}_n^t$. Since the clients with restricted resources may not implement full-precision model training, one may instead opt for quantization-aware training. In that case, the aggregation at the server can be described as

$$\begin{aligned} \mathbf{W}^t &= \sum_{k=1}^{N} p_n Q_{\mathbf{b}_n}(\mathbf{W}_n^t) \\ \text{s.t.} \quad & \mathbf{b}_n \cdot \mathbf{m} / \|\mathbf{m}\|_1 \le v_n, \forall n \in [N], \end{aligned} \tag{2}$$

where $Q_{\mathbf{b}_n}$ denotes the mixed-precision quantizer, $\mathbf{b}_n \in \mathbb{Z}^L$ denotes the bit-width assigned to each layer; $\mathbf{m} = \{M^{(1)}, \ldots, M^{(L)}\} \in \mathbb{Z}^L$ is the number of parameters in each layer of model, L is the number of layers, and v_n is the budget of the average bit-width for client n. Since in resource-heterogeneous FL v_n varies across clients, naive aggregation according to Eq. 2 may discard beneficial knowledge of high-precision models.

3.2. Binary Representation of Model Parameters

In the conventional deep neural networks (DNNs), the full-precision model parameters are typically stored in 32-bit floating-point form. Compared to the floating-point format, integer-arithmetic operations such as *mult*, *add* and *shift*, on model parameters in fixed-point representation are more efficient and hardware-friendly. Following studies [22, 41, 58, 61], we assume clients perform training using low-precision fixed-point quantization. A B-bit matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{C \times K}$ of the parameters in the l-th layer of the model can be represented in binary format with a ternary matrix $\mathbf{B}^{(l)} \in \{0,1\}^{B \times C \times K}$, a layer-wise scaling factor $s^{(l)}$ with floating-point value, and a layer-wise zero-point $z^{(l)} \in \mathbb{Z}^+$ as

$$\mathbf{W}_{j,k}^{(l)} = \frac{s^{(l)}}{2^B - 1} \left(\sum_{i=1}^{B} 2^{i-1} \mathbf{B}_{i,j,k}^{(l)} - z^{(l)} \right), \tag{3}$$

where $z^{(l)}$ is typically set to 2^{B-1} (signed integer); the scaling factor $s^{(l)}$ is updated at each training round according to the maximum absolute value of the parameters at the l-th layer. Therefore, the product between activation $\mathbf{A} \in \mathbb{R}^{K \times U}$ and $\mathbf{W}^{(l)}$ can be simplified by using *shift* and *add* according to

$$\begin{aligned} \mathbf{A}_{\cdot,u}^{\top} \cdot \mathbf{W}_{j,\cdot}^{(l)} &= \frac{s^{(l)}}{2^B - 1} \sum_{k=1}^{K} \mathbf{A}_{k,u} \left(\sum_{i=1}^{B} 2^{i-1} \mathbf{B}_{i,j,k}^{(l)} - z^{(l)} \right) \\ &= \frac{s^{(l)}}{2^B - 1} \left(\sum_{i=1}^{B} 2^{i-1} \mathbf{A}_{\cdot,u}^{\top} \cdot \mathbf{B}_{i,j,\cdot}^{(l)} - z^{(l)} \sum_{k=1}^{K} \mathbf{A}_{k,u} \right). \end{aligned} \tag{4}$$

Note that $\mathbf{B}^{(l)}$ consists of discrete values 0 and 1, which cannot be searched for via gradient descent. To optimize over these binary parameters, we adopt the straight-through estimator (STE) [3] as in [58]. STE enables a quantized network to forward pass intermediate signals using model parameters represented in fixed-point format (as shown in Eq. 3) while computing the gradients with continuous floating-point parameters as

$$\begin{aligned} &\textbf{Forward: } \mathbf{W}_{j,k}^{(l)} = \frac{s^{(l)}}{2^B - 1} \left(\sum_{i=1}^{B} 2^{i-1} \mathbf{B}_{i,j,k}^{(l)} - z^{(l)} \right) \\ &\textbf{Backward: } \frac{\partial \mathcal{L}}{\partial \mathbf{B}_{i,j,k}^{(l)}} = \frac{s^{(l)} 2^{i-1}}{2^B - 1} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{j,k}^{(l)}} \end{aligned} \tag{5}$$

where **Backward** pass follows from the chain rule. BSQ [58] relaxes the binary constraint and allows using floating-point values to update $\mathbf{B}_{i,j,k}^{(l)}$, which means that BSQ trains the networks with simulated quantization [22]. Different from BSQ, we adapt the WAGE [57] strategy and update binary parameters using integer operations via the *power-of-two* function $S(\cdot)$ defined as

$$S(x) = 2^{\lceil \log x \rfloor}, \tag{6}$$

where $S(x)$ returns the nearest power-of-two of x. Then we compute an update of $\mathbf{W}_{j,k}^{(l)}$ via gradient descent with step size η according to

$$\Delta \mathbf{W}_{j,k}^{(l)} = -\nu \cdot \frac{s^{(l)}}{2^B - 1} \sum_{i=1}^{B} 2^{q_i - 1}, \tag{7}$$

where $\nu \in \{-1, 1\}$ denotes the sign of $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{j,k}^{(l)}}$, and q_i is the power of $2^i \cdot S\left(\eta \left| \frac{\partial \mathcal{L}}{\partial \mathbf{B}_{i,j,k}^{(l)}} \right|\right)$. According to Eq. 5, q_i is strictly ascending, i.e., $q_i < q_j$ for $i < j$. Let $\mathcal{Q} = \{q_1, \ldots, q_B\}$ denote the set of q_i, and let $\max(\mathcal{Q})$ be the maximum element in $\mathcal{Q}$. On one hand, if $\max(\mathcal{Q}) > B$ the absolute value of $\Delta \mathbf{W}_{j,k}^{(l)}$ exceeds the scale $s^{(l)}$ and thus $\Delta \mathbf{W}_{j,k}^{(l)}$ needs to be clipped as

$$C(\Delta \mathbf{W}_{j,k}^{(l)}) = -\nu \cdot \frac{s^{(l)}}{2^B - 1} \sum_{i=1}^{B} 2^{i-1}. \tag{8}$$

Note that $\Delta \mathbf{W}_{j,k}^{(l)}$ cannot be represented by a fixed-point integer value if $q_i \leq 0 \in \mathcal{Q}$. Adapting the strategy of updating parameters with small-magnitude gradients in WAGE [57], $\Delta \mathbf{W}_{j,k}^{(l)}$ is converted to

$$\begin{aligned} C(\Delta \mathbf{W}_{j,k}^{(l)}) = &- \nu \cdot \frac{s^{(l)}}{2^B - 1} \sum_{i=1}^{B} 2^{i-1} \cdot \mathcal{I}\{i \in \mathcal{Q}\} \\ &- \nu \cdot \frac{s^{(l)}}{2^B - 1} \text{Bernoulli}\Big(\sum_{q_i \leq 0} 2^{q_i - 1}\Big), \end{aligned} \tag{9}$$

where $\mathcal{I}(\cdot)$ is an indicator, and $\text{Bernoulli}(\cdot)$ randomly samples decimal parts to either 0 or 1. When the magnitude of the gradient is small, the integer part of $\Delta \mathbf{W}_{j,k}^{(l)}$ is always 0, which impedes the update of the parameters. Due to the second term on the right hand side in Eq. 9, $\mathbf{W}_{j,k}^{(l)}$ is updated with the minimum step size even if the gradient is very small. After converting $\Delta \mathbf{W}_{j,k}^{(l)}$ to the fixed-point format, one can update the parameters according to

$$\mathbf{W}_{j,k}^{(l)} \leftarrow \text{Clipping}\left(\mathbf{W}_{j,k}^{(l)} + C(\Delta \mathbf{W}_{j,k}^{(l)}), \min^{(l)}, \max^{(l)}\right), \tag{10}$$

where $\min^{(l)} = -\frac{s^{(l)}}{2^B - 1} z^{(l)}$ denotes the minimum value of the parameters, and $\max^{(l)} = \frac{s^{(l)}}{2^B - 1}\left(2^B - 1 - z^{(l)}\right)$ is the maximum value of the parameters in the l-th layer. Since the updated $\mathbf{W}_{j,k}^{(l)}$ is in fixed-point format, updating binary representation $\mathbf{B}_{i,j,k}^{(l)}$ is straightforward.

3.3. Sparsity-Promoting Training

Sparsity-promoting regularizers including L1 (Lasso) and L2 (ridge) have been widely used to induce sparsity e.g. during feature selection. Following BSQ [58], we use group Lasso regularization [18] to promote obtaining highly sparse parameters and ensure stable convergence. The group Lasso regularizer for the parameters in the l-th layer is defined as

$$R_{\text{GL}}(\mathbf{B}^{(l)}) = \sum_{i=1}^{\mathbf{b}^{(l)}} \left\| \mathbf{B}_{i,\cdot,\cdot}^{(l)} \right\|_2, \tag{11}$$

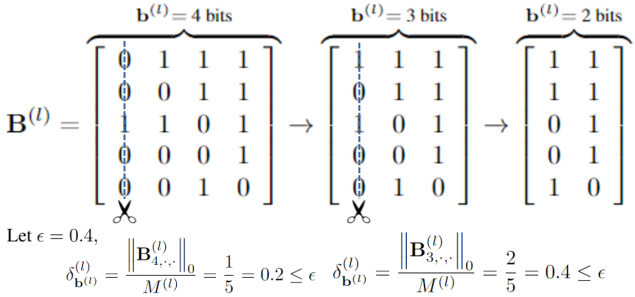


Figure 1. An example of pruning the MSBs in the model parameters $\mathbf{B}^{(l)}$. Since the 4-th and 3-rd bits in $\mathbf{B}^{(l)}$ have a fraction of 1's below the threshold ϵ, the bit-width $\mathbf{b}^{(l)}$ of the l-th layer is reduced from 4 bits to 2 bits. For clarity, ϵ is set to 0.4 in the example but the value of ϵ is smaller in the experiments.

where $\mathbf{b}^{(l)}$ denotes the bit-width of the l-th layer, and $\mathbf{B}_{i,\cdot,\cdot}^{(l)} \in \{0, 1\}^{C \times K}$ is the i-th binary representation position of the parameters in the l-th layer. Since the number of parameters typically differ from one layer to another, we vary the weights assigned to the group Lasso regularizers according to the memory constraints. Specifically, the objective function used in local training is formulated as

$$\mathcal{L}_{\text{local}} = \mathcal{L}_{\text{task}}(\mathbf{B}^{(1:L)}) + \lambda \sum_{l=1}^{L} \frac{M^{(l)}}{M} R_{\text{GL}}(\mathbf{B}^{(l)}), \tag{12}$$

where $M^{(l)}$ denotes the number of parameters in the l-th layer, $M = \sum_{l=1}^{L} M^{(l)}$, and λ is a non-negative predetermined hyper-parameter. Note that $\mathcal{L}_{\text{task}}(\mathbf{B}^{(1:L)})$ can be computed through the STE forward pass while the gradients can be computed through the STE backward pass as indicated in Eq. 5. Let $\delta_i^{(l)}$ denote the sparsity of the i-th binary representation position in the l-th layer,

$$\delta_i^{(l)} = \left\| \mathbf{B}_{i,\cdot,\cdot}^{(l)} \right\|_0 / M^{(l)}. \tag{13}$$

Essentially, $\delta_i^{(l)}$ is the proportion of the parameters having 1 at the i-th binary representation position. The smaller $\delta_i^{(l)}$ is, the higher the sparsity at the i-th position of the parameters' binary representation. We set up a threshold ϵ and prune the most significant bits (MSBs) if $\delta_{\mathbf{b}^{(l)}}^{(l)} \leq \epsilon$ (see Figure 1).

3.4. The End-to-End Training Procedure

The local sparsity-promoting training and MSBs pruning results in reduced bit-widths of the model parameters in certain layers, leading to a higher compression rate while maintaining the model's performance. However, our aim is to leverage mixed-precision quantization to improve the performance of the global model given all of the available resources, rather than pursue high compression rate. To this

end, we propose a *pruning-growing* procedure at the server that restores the average bit-width allocated to each local model. In particular, the server aggregates the collected local models characterized by different bit-width allocations into the global model $\mathbf{W}^{t+1}$ with the global bit-width allocation $\mathbf{b}^{t+1}$. For a given client, $\mathbf{b}^{t+1}$ is adjusted to satisfy the client's bit-width constraint via 2 operations: (1) *pruning*: reducing the bit-width of a subset of layers for the clients with low bit-width budget until the constraint in Eq. 2 is satisfied; (2) *growing*: increasing the bit-width of a subset of layers for the clients with high bit-width budget until violating the constraint in Eq. 2. Alg. 2 specifies the greedy policy used to select the layers for pruning or growing. Next, we provide an outline of the end-to-end training procedure, as formalized in Alg.1.

3.4.1 Initialization and Local Training

At the start of training, the local bit-width assignment of client n is initialized as $\hat{\mathbf{b}}_n^0 = \{v_n, \ldots, v_n\}$, while the global model $\mathbf{W}^0$ (full precision) is randomly initialized at the server. Then, $\mathbf{W}^0$ is converted to a v_n-bit fixed-point representation $\mathbf{G}_n^0$, with scale $\mathbf{s}_n^0 \in \mathbb{R}^L$ set to the double maximum absolute value of the model parameters in each layer, i.e.,

$$s_n^{(l),0} = 2 \cdot \max_{j,k} \left| \mathbf{W}_{j,k}^{(l),0} \right|. \tag{14}$$

The customized global model $\mathbf{G}_n^0$ is then sent to client n to be used for initialization of local training. With such an initialization, the local model $\mathbf{B}_n^0 \leftarrow \mathbf{G}_n^0$ satisfies the constraint of average bit-width v_n.

At global round t, client n receives the global model $\mathbf{G}_n^t$ and uses it to initialize $\mathbf{B}_n^t$. The client n updates $\mathbf{B}_n^t$, $\mathbf{b}_n^t$ and potentially $\mathbf{s}_n^t$, and possibly prunes MSBs of the model parameters across different layers as discussed in Sections 3.2 and 3.3. After updating, the client n sends $\mathbf{B}_n^{t+1}$ and $\mathbf{b}_n^{t+1}$ to the server for aggregation.

3.4.2 Models Aggregation

The server collects local models $\mathbf{B}_n^{t+1}$ and the corresponding local bit-width assignments $\mathbf{b}_n^{t+1}$ from the participating clients, $n \in [N]$, and then converts the collected fixed-point models to the floating-point models according to Eq. 3. The global model and the average bit-width are computed according to Eq. 2,

$$\mathbf{W}^{t+1} = \sum_{n=1}^{N} p_n \mathbf{W}_n^{t+1}, \quad \mathbf{b}^{t+1} = \sum_{n=1}^{N} p_n \mathbf{b}_n^{t+1}, \tag{15}$$

where $p_n = v_n \cdot |\mathcal{D}_n| / P$ is proportional to the number of samples in local dataset $\mathcal{D}_n$ and client n's budget v_n, and $P = \sum_{i=1}^{N} v_i \cdot |\mathcal{D}_i|$. Note that larger weights are assigned to the clients having higher bit-width budgets and training on larger local datasets.

Algorithm 1: FedMPQ

Input: global round T, local epochs τ, budget v_n, local data $\mathcal{D}_n$, number of parameters $\mathbf{m}$, λ and ϵ.
Output: the global model $\mathbf{W}^T$

initialize $\mathbf{W}^0$, $\mathbf{G}_n^0$, $\hat{\mathbf{b}}_n^0 \leftarrow \{v_n, \ldots, v_n\}, \forall n \in [N]$;
for *t = 0, …, T* **do**
  /* local training in the clients */
  for *n = 1, …, N* **do**
    $\mathbf{B}_n^t \leftarrow \mathbf{G}_n^t$, $\mathbf{b}_n^t \leftarrow \hat{\mathbf{b}}_n^t$;
    $\mathbf{B}_n^{t+1}, \mathbf{b}_n^{t+1} \leftarrow \text{LocalUpdate}(\mathcal{D}_n, \tau, \lambda, \epsilon)$;
    send $\mathbf{B}_n^{t+1}, \mathbf{b}_n^{t+1}$ to the server;
  end
  /* aggregation in the server */
  for *n = 1, …, N* **do**
    $\mathbf{W}_n^{t+1} \leftarrow \text{ConvertToFP}(\mathbf{B}_n^{t+1}, \mathbf{b}_n^{t+1})$;
  end
  $\mathbf{W}^{t+1} \leftarrow \sum_n^N p_n \mathbf{W}_n^{t+1}$, $\mathbf{b}^{t+1} \leftarrow \sum_n^N p_n \mathbf{b}_n^{t+1}$;
  /* post-aggregation adjustment */
  for *n = 1, …, N* **do**
    $\Delta \mathbf{b}_n^t \leftarrow \hat{\mathbf{b}}_n^t - \mathbf{b}_n^{t+1}$;
    $\hat{\mathbf{b}}_n^{t+1} \leftarrow$ **Pruning-Growing**$(\mathbf{b}^{t+1}, \Delta \mathbf{b}_n^t, \mathbf{m}, v_n)$;
    $\mathbf{G}_n^{t+1} \leftarrow \text{Binary-Representation}(\mathbf{W}^{t+1}, \hat{\mathbf{b}}_n^{t+1})$;
  end
end

The local bit-width assignment $\mathbf{b}_n^{t+1}$ is learned during the local sparsity-promoting training, where the bit-widths for less sensitive layers are reduced while the bit-widths for more sensitive layers are preserved. Note that the sensitivity of layers can vary across clients with non-i.i.d data, as it is affected by the data distribution. $\mathbf{b}^{t+1}$ aggregates local bit-width assignments and is reflective of the importance of different layers.

3.4.3 Post-Aggregation Adjustment

Due to the constraints on the local bit-width, the aggregated full-precision global model W^{t+1} cannot be directly broadcasted to the clients. Similar to the initialization described in Section 3.4.1, the server needs to customize different fixed-point global models $\mathbf{G}_n^{t+1}$ with bit-width assignments $\hat{\mathbf{b}}_n^{t+1}$ based on $\mathbf{b}^{t+1}$ and budget v_n. There are three options for updating $\mathbf{b}_n^{t+1}$:

$$\hat{\mathbf{b}}^{t+1} = \begin{cases} \mathbf{b}^{t+1}, \text{ if } v = v_n, \\ \text{pruning}(\mathbf{b}^{t+1}), \text{ if } v > v_n, \\ \text{growing}(\mathbf{b}^{t+1}), \text{ if } v < v_n, \end{cases} \tag{16}$$

where $v = \mathbf{b}^{t+1} \cdot \mathbf{m} / \|\mathbf{m}\|_1$. The first assignment is handled in a straightforward manner. For the later two we apply

Algorithm 2: Pruning-Growing

Input: $\mathbf{b}^{t+1}, \Delta\mathbf{b}_n^t, \mathbf{m}, v_n$
Output: bit-width assignment for client n, $\hat{\mathbf{b}}_n^{t+1}$

initial: $\hat{\mathbf{b}}_n^{t+1} \leftarrow \mathbf{b}^{t+1}$, $v \leftarrow \hat{\mathbf{b}}_n^{t+1} \cdot \mathbf{m} / \|\mathbf{m}\|_1$, $\mathbf{d} \leftarrow \text{argDescending}(\mathbf{m} \odot (\Delta\mathbf{b}_n^t + \mathbf{1}))$, $cur \leftarrow 0$;
/* Pruning */
while $v > v_n$ ***and*** $cur < |\mathbf{m}|$ **do**
  $l \leftarrow \mathbf{d}[cur]$;
  if $\hat{\mathbf{b}}_n^{t+1}[l] > 1$ **then**
    $\hat{\mathbf{b}}_n^{t+1}[l] \leftarrow \hat{\mathbf{b}}_n^{t+1}[l] - 1$, $v \leftarrow v - \mathbf{m}[l] / \|\mathbf{m}\|_1$;
  else
    $cur \leftarrow cur + 1$;
  end
end
$cur \leftarrow |\mathbf{m}| - 1$;
/* Growing */
while $v < v_n$ ***and*** $cur > 0$ **do**
  $l \leftarrow \mathbf{d}[cur]$;
  if $\hat{\mathbf{b}}_n^{t+1}[l] < 8$ **then**
    $\hat{\mathbf{b}}_n^{t+1}[l] \leftarrow \hat{\mathbf{b}}_n^{t+1}[l] + 1$, $v \leftarrow v + \mathbf{m}[l] / \|\mathbf{m}\|_1$;
  else
    $cur \leftarrow cur - 1$;
  end
end

a *greedy* policy for executing pruning or growing. Specifically, when $v > v_n$, we attempt to reduce the bit-width of the layer having the most parameters, attempting to satisfy the budget constraint v_n while maintaining the precision of other layers. When $v < v_n$, we first increase the bit-width of the layer with the fewest parameters. If two layers have the same number of parameters, the pruning preference is given to the layer whose bit-width in the last round of local sparsity-promoting training had been reduced more, as that suggests this layer is less important. It is beneficial to record the bit-width change during local training by defining $\Delta\mathbf{b}_n^t = \hat{\mathbf{b}}_n^t - \mathbf{b}_n^{t+1}$. Further details of pruning and growing are specified in Alg. 2.

4. Experiments

4.1. Setup

We evaluate the performance of the proposed FedMPQ in various settings on three datasets: CIFAR10, CIFAR100 and Tiny-ImageNet [28]. We perform the experiments using ResNet20 [19] model for CIFAR10/100, and ResNet44 [19] model for Tiny-ImageNet. We use the mini-batch stochastic gradient descent (SGD) with a learning rate initialized to 0.1 in all experiments. The SGD momentum is set to 0.9 and the weight decay is set to 0.0005. The batch size is set to 64 and the number of global rounds is set to 50, with 5 local epochs within each global round. The value of the bit-pruning threshold ϵ is 0.03, while the regularizing hyper-parameter λ is equal to 0.01. The number of clients is 10; unless stated otherwise, the fraction of participating clients is 0.5. Following the strategy in [63], we use Dirichlet distribution with varying concentration parameter α to generate data partitions at different levels of heterogeneity (smaller α leads to generating less balanced data).

Since there exist no prior methods for mixed-precision quantization-aware training (QAT) in FL, we primarily focus on comparing FedMPQ with the AQFL [1], the first method to deploy fixed-precision quantization-aware training in FL. As FedMPQ only modifies the precision of the model weights, we fix the precision of the activation to 4 bits in both FedMPQ and AQFL throughout the entire training process. Furthermore, we implement FedAvg with full-precision (FP32) training, 8-bits fixed-precision quantization-aware training (FPQ8) [22], and two communication-efficient FL methods, FedPAQ [39] and UVeQFed [43], which train full-precision local models but then utilize scalar and vector quantization to compress the local updates before communicating them to the server.

4.2. Effects of Data Heterogeneity

To evaluate our method in the scenarios characterized by varied levels of data heterogeneity, we conduct 3 sets of experiments where α takes on values from $\{0.1, 0.5, 1\}$; these correspond to severely imbalanced, moderately imbalanced and mildly imbalanced data, respectively. Results of the experiments are reported in Table 1. As can be seen there, performance of the global model of all the considered methods deteriorates as the data heterogeneity increases. FedPAQ and UVeQFed, the two communication-efficient FL approaches, achieve performance comparable to the FP32 baseline when $\alpha = 0.5$ and 1 but experience performance degradation when $\alpha = 0.1$. A significant performance decline is experienced in the experiments with FPQ8 due to the low capacity of the low-precision models. However, the performance gap between FP32 and FPQ8 narrows as the level of data heterogeneity increases. For instance, when $\alpha = 1$, the test accuracy of FP32 is 11.3% higher than that of FPQ8; when $\alpha = 0.1$, the test accuracy difference is 5.8% (experiments on CIFAR10). When the data is extremely imbalanced, FedAvg suffers from the so-called "client-drift" problem [27] caused by overfitting on the local data. As a result, the model capacity advantage of FP32 (due to having higher precision) might not make as much of a positive impact on the accuracy, explaining the narrowing gap between FP32 and FPQ8.

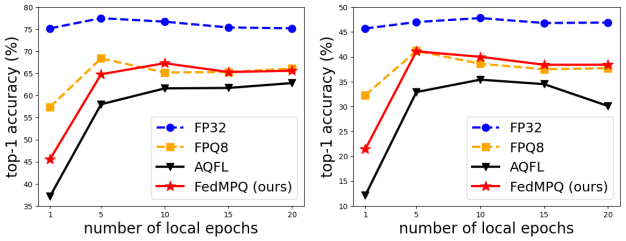
Since local models with ultra-low precision are aggregated into the global model, performance of AQFL shows further deterioration. The proposed method, FedMPQ outperforms AQFL (implementing fixed-precision quantization) in all scenarios even though training models under the same resource constraints. As shown in Table.1, FedMPQ

Table 1. Test accuracy (%) of the considered schemes as the concentration parameter α takes values from $\{0.1, 0.5, 1\}$. The number of clients in these experiments is 10, while their average bit-width budgets are $\mathbf{v} = \{2, 2, 4, 4, 4, 6, 6, 8, 8, 8\}$ (v_n denotes the budget of client n). The numbers in the column "Update", "Weight" and "Activation" indicate the bits used to store the local update values, model weights and the activation signals, respectively. The last column indicates whether the scheme needs to train a full-precision model or not.

	CIFAR10			CIFAR100			Tiny-ImageNet			Update	Weight	Activation	Full-Precision?
α	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1				
FP32	60.3	77.5	82.1	40.3	47.0	49.6	24.6	35.1	38.1	32	32	32	✔
FedPAQ	56.3	77.0	81.2	39.7	46.8	48.4	22.87	34.6	37.5	v_n	32	32	✔
UVeQFed	56.8	76.7	81.5	38.6	46.5	48.8	21.3	34.3	37.4	v_n	32	32	✔
FPQ8	54.5	68.4	70.8	35.4	41.3	42.3	23.8	33.4	35.6	8	8	8	✗
AQFL	44.3	58.0	62.1	23.8	32.9	36.1	17.1	23.5	25.3	v_n	v_n	4	✗
FedMPQ	**49.1**	**67.1**	**69.3**	**31.7**	**41.1**	**43.6**	**20.3**	**27.0**	**28.2**	v_n	v_n	4	✗

Table 2. Test accuracy (%) as the number of clients N varies over 10, 20 and 40. Here, 20%, 30%, 30% and 20% clients have average bit-width budget of 2 bits, 4 bits, 6 bits and 8 bits, respectively. The concentration parameter α is set to 0.5.

	CIFAR10			CIFAR100		
N	10	20	40	10	20	40
FP32	77.5	68.1	64.5	47.0	40.1	35.0
FedPAQ	77.0	61.5	59.9	46.8	38.9	33.1
UVeQFed	76.7	66.4	63.2	46.5	39.5	34.1
FPQ8	68.4	56.3	48.4	41.3	36.2	31.9
AQFL	58.0	49.7	37.3	32.9	20.4	13.9
FedMPQ	**67.1**	**56.8**	**45.4**	**41.1**	**26.1**	**19.9**



(a) CIFAR10 (b) CIFAR100

Figure 2. Top-1 accuracy vs. the number of local epochs. In all experiments, the number of clients is $N = 10$, the concentration parameter $\alpha = 0.5$, and the number of global rounds is set to 50. All results are the final test accuracy after 50 global rounds.

outperforms AFQL at most 9.1%, 8.2% and 2.9% test accuracy on CIFAR10, CIFAR100 and Tiny-ImageNet respectively. On CIFAR10/100 datasets, FedMPQ nearly preserves performance of FPQ8 baseline in the settings $\alpha = 0.5$ and 1 by efficiently allocating precision to different layers, even though training the global model on resource-constrained heterogeneous devices.

4.3. Scalability

To evaluate the effect of the system size on the performance of FedMPQ, we conducted 3 sets of experiments on CIFAR10/100 data in the FL system with 10, 20 and 40 clients. To simulate a resource-constrained heterogeneous system, we allocate to 20%, 20%, 20% and 30% clients the bit-width budget of 2, 4, 6 and 8 bits, respectively. The concentration parameter α is set to 0.5 and kept constant throughout the experiments.

Not surprisingly, as the results in Table 2 show, the performance of all schemes deteriorates as the FL training involves an increasingly larger number of clients with ultra-low budget. FedMPQ consistently outperforms AQFL, with up to 9.1% and 8.2% higher accuracy on CIFAR10 and CIFAR100, respectively. Note that FedMPQ manages to approach the performance of FPQ8 in the settings involving 10 clients, though the gap (as expected) widens as the number of clients participating in the system grows.

4.4. The Number of Local Epochs

We study the impact of the number of local epochs on the system's performance by conducting 5 sets of experiments where the number of local epochs is varied over $\{1, 5, 10, 15, 20\}$. The results are shown in Figure 2. As can be seen there, when the number of local epochs is set to 1, the three quantization-aware training methods show significant performance degradation while the impact on FP32 baseline is only minor. This is likely due to underfitting of the quantized models. For larger numbers of local epochs, FedMPQ provides a global model whose performance is comparable to FPQ8, demonstrating remarkable improvement over AQFL.

4.5. Fine-Tuning Hyper-Parameters

The experiments discussed in the previous section are conducted with hyperparameters $\epsilon = 0.03$ and $\lambda = 0.01$. Note that ϵ and λ jointly affect the training: λ controls the weight of the group Lasso regularization used in local training while ϵ controls the threshold for pruning the

Table 3. Test accuracy (%) of FedMPQ running with different combinations of threshold value ϵ and regularization weights λ. All experiments are on CIFAR10 with 10 clients and $\alpha = 0.5$.

	threshold value ϵ				
λ	0.01	0.02	0.03	0.04	0.05
0.1	64.1	63.4	65.9	63.9	64.8
0.01	66.4	68.0	67.1	64.7	64.3
0.001	67.1	70.0	65.5	64.7	64.4

MSBs after local training. To explore the space of hyper-parameters, we consider a number of configurations with varied values of $\epsilon \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$ and $\lambda \in \{0.1, 0.01, 0.001\}$. As the results shown in Table 3 indicate, when the threshold ϵ is too large, the accuracy considerably deteriorates since a larger fraction of model parameters gets compressed. Selecting large regularization weights, e.g. $\lambda = 0.1$, leads to the performance drop since FedMPQ focuses on pursuing higher bit-level sparsity. Our experiments suggest that to achieve satisfactory performance, one should select $\epsilon \leq 0.03$ and $\lambda \leq 0.01$.

4.6. An Ablation Study

In this section, we empirically analyze the effect of each procedure in FedMPQ by a comparison to the AQFL baseline. The three procedures that distinguish FedMPQ from AQFL include: (1) the group Lasso regularization in the objective function as described in Section 3.3; (2) bit-level pruning in the most significant bits (MSBs); and (3) Alg. 2 which restores the precision of local models. We refer to different combinations of these procedures, shown in Table 4, as settings (1)-(5).

According to the results for setting (1) in Table 4, the group Lasso regularization achieves high performance while promoting bit-level sparsity, with small 1.2% and 0.3% drops in accuracy on CIFAR10 and CIFAR100, respectively. As shown in the results for setting (2), MSBs pruning without sparsity-promoting training causes more severe performance degradation, with 2.9% and 4.2% accuracy drop on these two datasets. Finally, by combining the group Lasso regularization with MSBs pruning (setting (3)) enables the global model to achieve performance close to the baseline, even though the precision is reduced after MSBs pruning. Interestingly, the group Lasso regularization forces the clients to learn local models with highly sparse binary weight representations, implying no degradation due to bit-pruning.

Algorithm 2 enables clients to recover layer-wise precision budget allocated to their local models, resulting in significant performance improvements. For instance, test accuracy in setting (4) is 10.3% and 11.4% higher than in setting (2); setting (5) achieves 10.9% and 9.3% higher accuracy than setting (3) on CIFAR10 and CIFAR100, respectively.

Table 4. Test accuracy (%) of the global model trained using different combinations of the FedMPQ subroutines. "Lasso" refers to the group Lasso regularization; "MSBs" denotes bit-level pruning. All experiments involve 10 clients. α, ϵ and λ are set to 0.5, 0.03 and 0.01, respectively (the same setting as in Section 4.1).

	CIFAR10	**CIFAR100**
AQFL (baseline)	58.0	32.9
(1) Lasso	56.8	32.6
(2) MSBs	55.1	28.7
(3) MSBs + Lasso	56.2	31.8
(4) MSBs + Alg.2	65.4	40.1
(5) MSBs + Lasso + Alg.2	**67.1**	**41.1**

This ablation study provides an insight in how FedMPQ operates: the group Lasso regularization forces clients to learn a local model with bit-level sparsity while preserving performance; MSBs pruning allows reduction of the precision of local models without major performance degradation; Algorithm 2, implemented at the server, conducts *pruning-growing* to restore the precision of local models so they fully exploit allocated bit-width budgets while seeking effective bit-width allocations to layers. The results in Table 4 suggest that Algorithm 2 plays a major role in helping FedMPQ achieve high accuracy.

5. Conclusion

In this paper we presented FedMPQ, a novel framework for heterogeneous, resource-constrained federated learning systems, which aims to judiciously utilize bit-width budget of clients by conducting mixed-precision quantization. A group Lasso regularizer applied in local training promotes sparsity of the binary representation of the model parameters, and then reduces the precision of a subset of the local model's layers. The server deploys a greedy *pruning-growing* procedure that restores precision of the pruned local models to fully exploit the assigned bit-width budget. We conducted extensive experiments on several benchmark datasets for a number of problem configurations that simulate resource-heterogeneity across clients. The experimental results demonstrate that FedMPQ outperforms the fixed-precision quantization baseline, and provides performance remarkably close to the 8-bit quantization baseline.

6. Acknowledgements

This material is based upon work supported in part by the National Science Foundation under grant 2148224.

References

[1] Ahmed M Abdelmoniem and Marco Canini. Towards mitigating device heterogeneity in federated learning via adaptive model quantization. In *Proceedings of the 1st Workshop on Machine Learning and Systems*, pages 96–103, 2021. 1, 2, 6

[2] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021. 1

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2, 3

[4] Zhaowei Cai and Nuno Vasconcelos. Rethinking differentiable search for mixed-precision neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2349–2358, 2020. 2

[5] Huancheng Chen and Haris Vikalo. Federated learning in non-iid settings aided by differentially private synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5026–5035, 2023. 1

[6] Huancheng Chen and Haris Vikalo. Accelerating non-iid federated learning via heterogeneity-guided client sampling. *arXiv preprint arXiv:2310.00198*, 2023.

[7] Huancheng Chen, Chaining Wang, and Haris Vikalo. The best of both worlds: Accurate global and personalized models through federated learning with data-free hyper-knowledge distillation. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[8] Shengbo Chen, Cong Shen, Lanxue Zhang, and Yuanmin Tang. Dynamic aggregation for heterogeneous quantization in federated learning. *IEEE Transactions on Wireless Communications*, 20(10):6804–6819, 2021. 2

[9] Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5350–5359, 2021. 3

[10] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 1

[11] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 1, 3

[12] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020. 3

[13] Ahmed Roushdy Elkordy and A Salman Avestimehr. Heterosag: Secure aggregation with heterogeneous quantization in federated learning. *IEEE Transactions on Communications*, 70(4):2372–2386, 2022. 1, 2

[14] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 544–560. Springer, 2020. 2

[15] Hai Victor Habi, Roy H Jennings, and Arnon Netzer. Hmq: Hardware friendly mixed precision quantization block for cnns. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pages 448–463. Springer, 2020. 2

[16] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 2350–2358. PMLR, 2021. 2

[17] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 2350–2358. PMLR, 2021. 1

[18] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015. 4

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[20] Robert Hönig, Yiren Zhao, and Robert Mullins. Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning. In *International Conference on Machine Learning*, pages 8852–8866. PMLR, 2022. 1, 2

[21] Xijie Huang, Zhiqiang Shen, Shichao Li, Zechun Liu, Hu Xianghong, Jeffry Wicaksana, Eric Xing, and Kwang-Ting Cheng. Sdq: Stochastic differentiable quantization with mixed precision. In *International Conference on Machine Learning*, pages 9295–9309. PMLR, 2022. 2

[22] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 3, 6

[23] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 2

[24] Divyansh Jhunjhunwala, Advait Gadhikar, Gauri Joshi, and Yonina C Eldar. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3110–3114. IEEE, 2021. 1, 2

[25] Yu Ji and Lan Chen. Fedqnn: A computation–communication-efficient federated learning framework for iot with low-bitwidth neural network quantization. *IEEE Internet of Things Journal*, 10(3):2494–2507, 2022. 1

[26] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista

Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 1

[27] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020. 1, 6

[28] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 6

[29] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 2

[30] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019. 1

[31] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[32] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020. 1

[33] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019. 1

[34] Qian Lou, Feng Guo, Lantao Liu, Minje Kim, and Lei Jiang. Autoq: Automated kernel-wise neural network quantization. *arXiv preprint arXiv:1902.05690*, 2019. 2

[35] Yuexiao Ma, Taisong Jin, Xiawu Zheng, Yan Wang, Huixia Li, Yongjian Wu, Guannan Jiang, Wei Zhang, and Rongrong Ji. Ompq: Orthogonal mixed precision quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9029–9037, 2023. 3

[36] Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4): 1–26, 2022. 2

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 3

[38] Kaan Ozkara, Navjot Singh, Deepesh Data, and Suhas Diggavi. Quped: Quantized personalization via distillation with applications to federated learning. *Advances in Neural Information Processing Systems*, 34:3622–3634, 2021. 1

[39] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020. 1, 2, 6

[40] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019. 1

[41] Yusuke Sekikawa and Shingo Yashima. Bit-pruning: A sparse multiplication-less dot-product. In *The Eleventh International Conference on Learning Representations*, 2022. 3

[42] Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. Federated learning with quantization constraints. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8851–8855. IEEE, 2020. 1, 2

[43] Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. Uveqfed: Universal vector quantization for federated learning. *IEEE Transactions on Signal Processing*, 69:500–514, 2020. 1, 2, 6

[44] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018. 1

[45] Zhenhong Sun, Ce Ge, Junyan Wang, Ming Lin, Hesen Chen, Hao Li, and Xiuyu Sun. Entropy-driven mixed-precision quantization for deep network design. *Advances in Neural Information Processing Systems*, 35:21508–21520, 2022. 3

[46] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2

[47] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020. 1

[48] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 1

[49] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020. 3

[50] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *European Conference on Computer Vision*, pages 259–275. Springer, 2022. 3

[51] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. *arXiv preprint arXiv:1905.11452*, 2019. 1, 2

[52] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020. 1

[53] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019. 1, 2

[54] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019. 1

[55] Blake E Woodworth, Jialei Wang, Adam Smith, Brendan McMahan, and Nati Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. *Advances in neural information processing systems*, 31, 2018. 1

[56] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018. 1, 2

[57] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*, 2018. 3, 4

[58] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021. 1, 3, 4

[59] Linjie Yang and Qing Jin. Fracbits: Mixed precision quantization via fractional bit-widths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10612–10620, 2021. 1, 2

[60] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021. 3

[61] Jaehong Yoon, Geon Park, Wonyong Jeong, and Sung Ju Hwang. Bitwidth heterogeneous federated learning with progressive weight dequantization. In *International Conference on Machine Learning*, pages 25552–25565. PMLR, 2022. 1, 2, 3

[62] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5693–5700, 2019. 1

[63] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pages 7252–7261. PMLR, 2019. 6

[64] Zhaoyang Zhang, Wenqi Shao, Jinwei Gu, Xiaogang Wang, and Ping Luo. Differentiable dynamic quantization with mixed precision and adaptive resolution. In *International Conference on Machine Learning*, pages 12546–12556. PMLR, 2021. 1, 2

[65] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021. 1