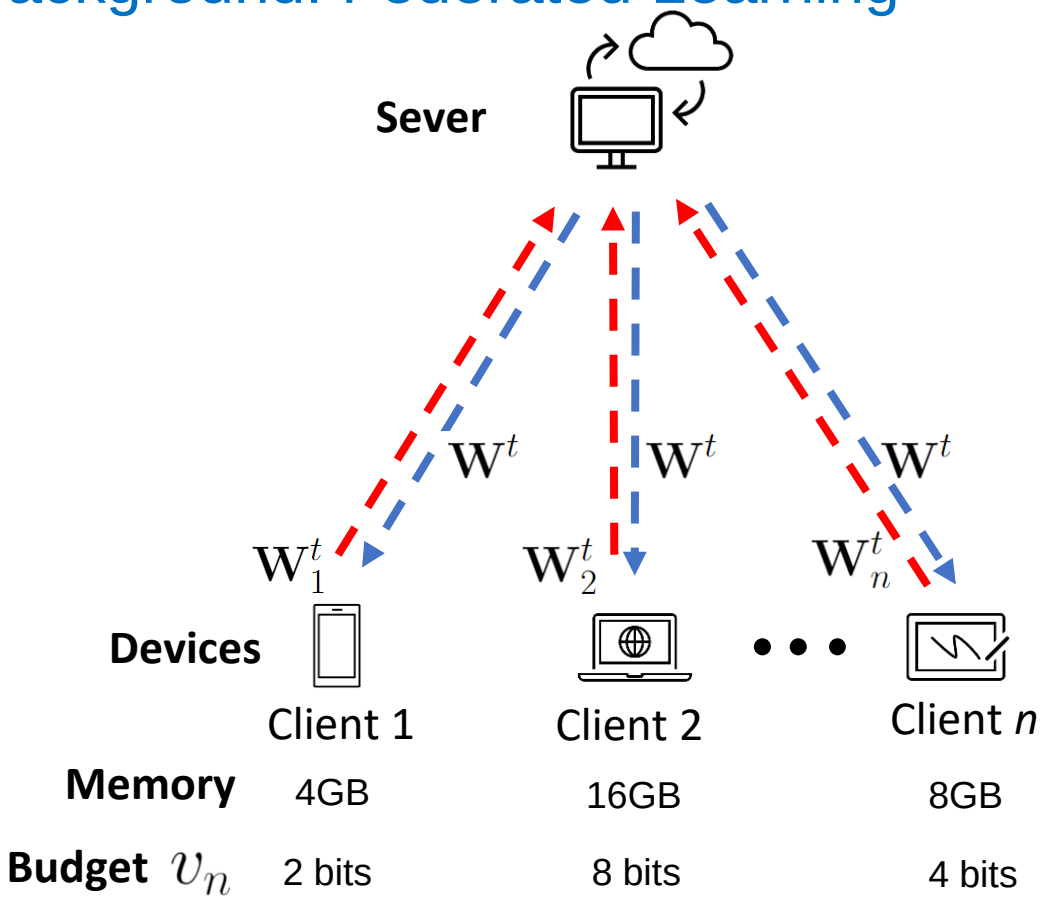
Mixed-Precision Quantization for Federated Learning on Resource-Constrained Heterogeneous Devices

Huancheng Chen, Haris Vikalo

University of Texas at Austin

Background: Federated Learning



We consider resource-constrained settings where a quantized aggregated model is formed as

$$\mathbf{W}^t = \sum_{k=1}^{N} p_n Q_{\mathbf{b}_n}(\mathbf{W}_n^t)$$
$$\text{s.t.} \quad \mathbf{b}_n \cdot \mathbf{m} / \|\mathbf{m}\|_1 \leq \upsilon_n, \forall n \in [N],$$

where $Q_{\mathbf{b}_n}$ denotes the mixed-precision quantizer, $\mathbf{b}_n \in \mathbb{Z}^L$ is 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, υ_n is the budget of the average bit-width for client *n*.

The Challenges:

- Reinforcement learning (RL) and neural architecture search (NAS) are not feasible on resource-constrained devices.
- Devices have no access to pretrained full-precision models.
- FL requires quantization-aware training (QAT) rather than pos-training quantization (PTQ).

Binary Representation of Model Parameters

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 using 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

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

Sparsity-Promoting Training

The objective function used in local training is formulated as

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

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

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 pre-determined hyper-parameter.

Straight-Through Estimator (STE)

STE enables a quantized network to forward pass intermediate signals using model parameters represented in fixed-point format while computing the gradients with continuous floating-point parameters in the backward pass as follows

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

$$\textbf{Backward:}\ \frac{\partial \mathcal{L}}{\partial \mathbf{B}_{i,j,k}^{(l)}} = \frac{s^{(l)}2^{i-1}}{2^B - 1}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{j,k}^{(l)}}$$

As the gradients computed from backward pass are floating-point, we convert the weight updates $\triangle \mathbf{W}_{j,k}^{(l)}$ into integers via the *power-of-two* function defined as

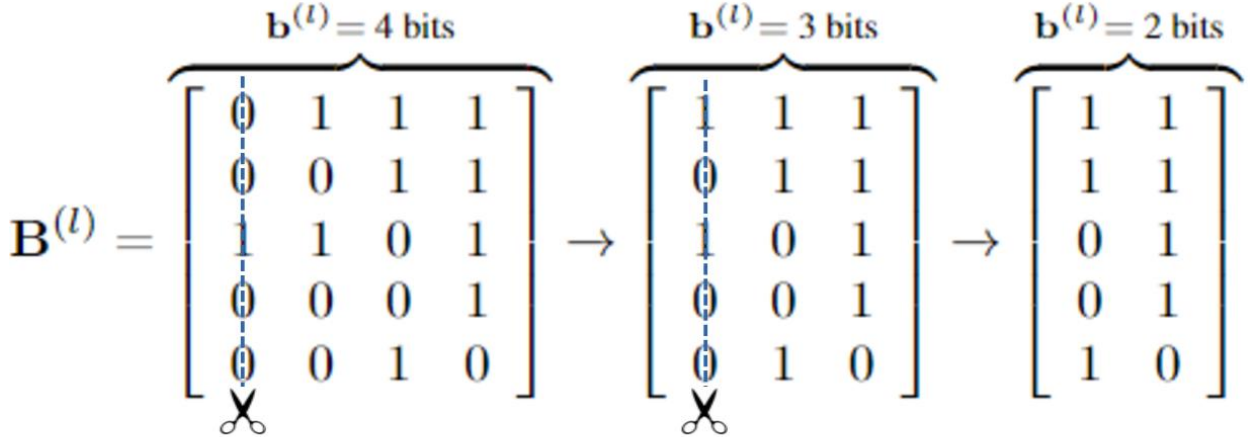
$$S(x) = 2^{\lceil \log x \rfloor}$$

Bit-level Pruning

The sparsity of the *i*-th binary representation position in the *l*-th layer:

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

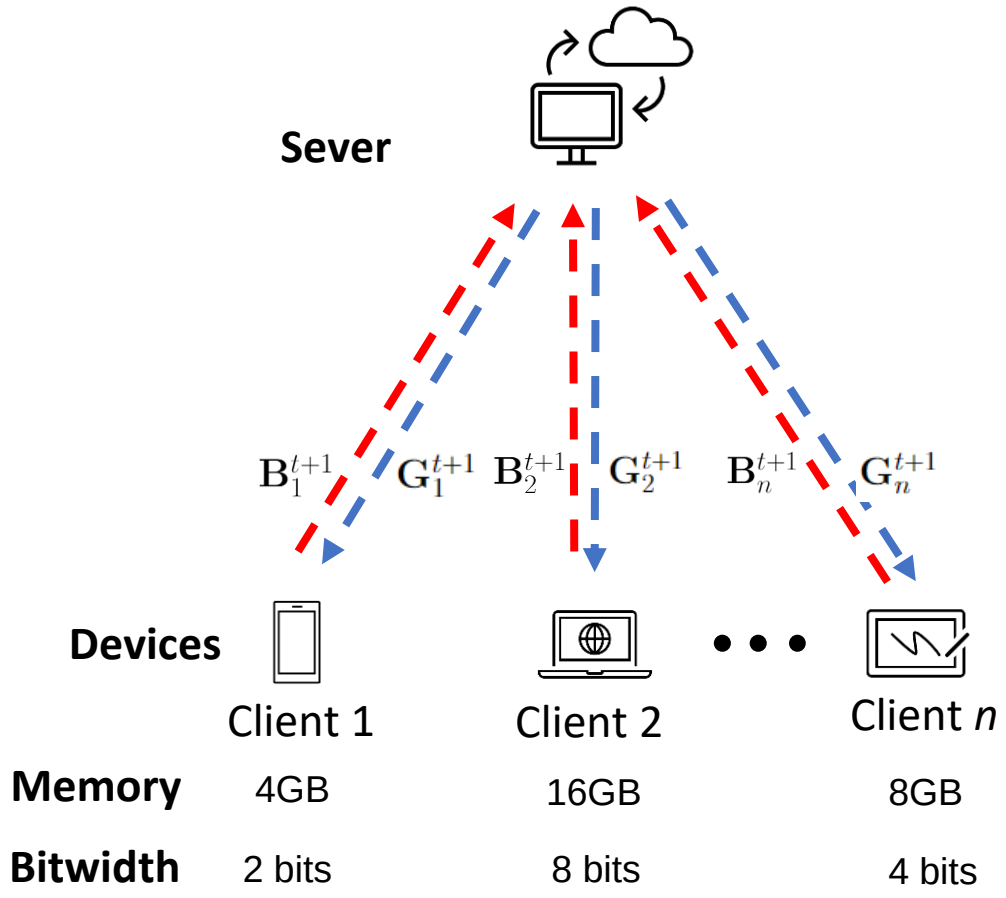
An example of pruning the MSBs in the model parameters $\mathbf{B}^{(l)}$:



Let $\epsilon = 0.4$,

$$\delta_{\mathbf{b}^{(l)}}^{(l)} = \frac{\left\| \mathbf{B}_{4,\cdot,\cdot}^{(l)} \right\|_0}{M^{(l)}} = \frac{1}{5} = 0.2 \leq \epsilon \quad \delta_{\mathbf{b}^{(l)}}^{(l)} = \frac{\left\| \mathbf{B}_{3,\cdot,\cdot}^{(l)} \right\|_0}{M^{(l)}} = \frac{2}{5} = 0.4 \leq \epsilon$$

Overall Procedure



Step 1: each client implement local update

$$\mathbf{B}_n^{t+1}, \mathbf{b}_n^{t+1} \leftarrow \text{LocalUpdate}(\mathcal{D}_n, \tau, \lambda, \epsilon)$$

Step 2: the sever collects $\mathbf{B}_n^{t+1}$ and converts into FP

$$\mathbf{W}_n^{t+1} \leftarrow \text{ConvertToFP}(\mathbf{B}_n^{t+1}, \mathbf{b}_n^{t+1})$$

Step 3: the server computes weighted average

$$\mathbf{W}^{t+1} \leftarrow \sum_n^N p_n \mathbf{W}_n^{t+1},\ \mathbf{b}^{t+1} \leftarrow \sum_n^N p_n \mathbf{b}_n^{t+1}$$

Step 4: the server computes optimal bit-width for each client and converts FP into bit-presentation

$$\Delta \mathbf{b}_n^t \leftarrow \hat{\mathbf{b}}_n^t - \mathbf{b}_n^{t+1}$$

$$\hat{\mathbf{b}}_n^{t+1} \leftarrow \textbf{Pruning-Growing}(\mathbf{b}^{t+1}, \Delta \mathbf{b}_n^t, \mathbf{m}, v_n)$$

$$\mathbf{G}_n^{t+1} \leftarrow \text{Binary-Representation}(\mathbf{W}^{t+1}, \hat{\mathbf{b}}_n^{t+1})$$

Experimental Settings

- **Data partitions:** Clients' data is generated from the Dirichlet distribution with concentration parameter **α**. The proportion p_c of samples with label ***c*** among ***N*** clients is drawn as

$$p_c \sim \mathbf{Dir}_N(\alpha)$$

- **Bit-width budgets:** The number of clients in these experiments is 10, while their average bit-width budgets are **v** = {2,2,4,4,4,6,6,6,8,8}.

- **Dataset:** CIFAR10, CIFAR100, Tiny-ImageNet

Experimental Results

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 $\mathbf{v} = \{2, 2, 4, 4, 4, 6, 6, 6, 8, 8\}$ (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	✗