A DETAILED EXPERIMENTAL SETTINGS

A.1 MODELS AND DATASETS

In this paper, we use a lightweight version VGG11 for SVHN. Specifically, we halve the output channels of its convolutional layers and the number of output nodes in its fully connected layers. The implementations of ResNet-18 and ResNet-34 follow the same structure as described in the original paper. ReLU is used as the activation function, and batch normalization (BN) Ioffe & Szegedy (2015) is also used to improve the stability of training. For each dataset, we split the training set to 10 clients and use the test set to verify the accuracy of the aggregated global model on the server.

A.2 BASELINES

In this part, we introduce the implementation and tuning of the baseline methods in as much detail as possible. The tuning results are shown in Table 2.

SignSGD only transfers the signs of the model updates and aggregates the signs by majority vote. Specifically, for a parameter x, $x^{t+1} = x^t + \eta \cdot \text{sign}(\sum_{k \in \mathbb{C}^t} \text{sign}(\Delta x_k^{t+1}))$, where Δx_k^{t+1} is the parameter update from the k_{th} client and $\mathbb{C}^t$ is the set of selected clients. We tune the hyperparameter η in (0.0001, 0.0005, 0.001) to achieve best test accuracy.

FedPAQ is to quantize the model updates by a given quantization bitwidth while FedDQ is to quantize the model updates with a given step size. We tune the bitwidth in (2,4,8) for FedPAQ and the step size in (0.02, 0.01, 0.005) for FedDQ to ensure similar accuracy to FedAvg.

DAdaQuant dynamically adjusts the bitwidth used by each client in each round by monitoring the local training loss and the local dataset size. DAdaQuant consists of two parts, time-adaptive quantization and client-adaptive quantization. We first denote the set of selected clients in the t_{th} round as $\mathbb{C}^t$, the proportion of the k_{th} client's data quantity to the total data quantity as p_k and the training loss of the k_{th} client as l_k. In time-adaptive quantization, server tracks a running average loss $F^t = \psi F^{t-1} + (1-\psi)G^t$, where $G^t = \sum_{k \in \mathbb{C}^t} p_k \cdot l_k^t$. The server determines training to converge whenever $F^t \geq F^{t+1-\phi}$. On convergence, the quantization bitwidth will be doubled and then fixed for at least ϕ rounds. Referring to the original paper, we set ψ to 0.9, ϕ to 1/10 of the number of rounds. Also, the quantization bitwidth starts at 1 and will not exceed 4. In client-adaptive quantization, assuming the bitwidth of time-adaptive quantization is m, the bitwidth of the k_{th} client will be set to $p_k^{2/3} \cdot \sqrt{\sum_{k \in \mathbb{C}^t} p_k^{2/3} / \sum_{k \in \mathbb{C}^t} p_k^2 / m^2}$. We do not consider extra encoding compression techniques used in DAdaQuant since it is not required for nor part of quantization.

Table 2: Hyperparameters of baseline methods used for different models.

	SignSGD	FedPAQ	FedDQ	DAdaQuant	
	η	bitwidth	step size	ψ	ϕ
SVHN	0.0001	4	0.005	0.9	10
CIFAR-10	0.0001	4	0.01	0.9	20
CIFAR-100	0.0005	4	0.01	0.9	20

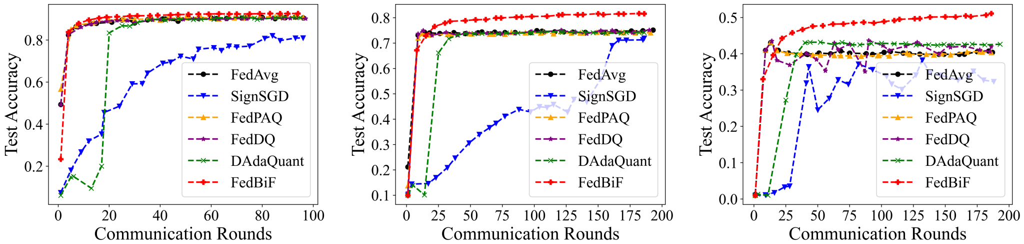
B ADDITIONAL EXPERIMENT RESULTS

Table 3: The average sparsity of VGG11, ResNet-18 and ResNet-34 trained by FedBiF.

Models	IID	Non-IID ($\mu = 0.5$)	Non-IID ($\mu = 0.1$)
VGG11	24.7%	24.1%	24.0%
ResNet-18	34.1%	26.0%	24.6%
ResNet-34	33.8%	30.4%	28.9%

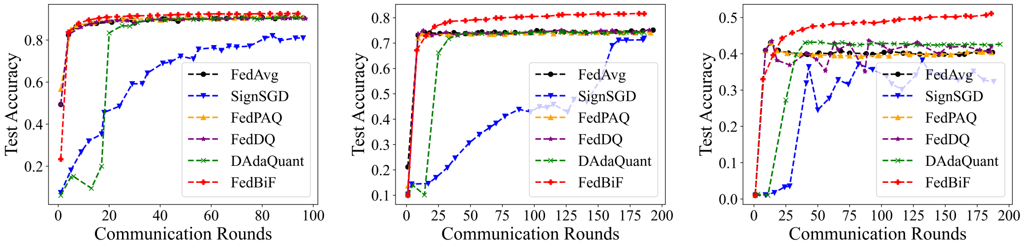
B.1 CONVERGENCE CURVES

As shown in Figure 7 and 8, we report the convergence curves of all methods on Non-IID datasets. In both Non-IID settings, the experimental results and conclusions are similar to those of the IID setting.



(a) SVHN (b) CIFAR-10 (c) CIFAR-100

Figure 7: Convergence curves of baseline methods and FedBiF on Non-IID ($\mu = 0.5$) datasets.



(a) SVHN (b) CIFAR-10 (c) CIFAR-100

Figure 8: Convergence curves of baseline methods and FedBiF on Non-IID ($\mu = 0.1$) datasets.

B.2 MODEL SPARSITY

As shown in Table 3, we report the average sparsity of each model trained by FedBiF. We find that higher sparsity is generally achieved with larger models and more even data distribution. It is quite intuitive, as larger models often tend to have more redundant parameters, whereas simpler data distributions typically require fewer parameters for effective learning.

C LIPSCHITZ CONTINUTY

In this paper, we consider the Lipschitz continuity to verify the generalization of the trained models.

Definition 1. *A function $f : \mathbb{R}^m \to \mathbb{R}^1$ is Lipschitz continuous on $\mathbb{X} \subset \mathbb{R}^m$, if there exists a constant $L \geq 0$ and a distance metric d that satisfies:*

$$|f(X_i) - f(X_j)| \leq L \cdot d(X_i, X_j), \forall X_i, X_j \in \mathbb{X} \tag{7}$$

Generally, norm is used as the distance metric, i.e., $d(X_i, X_j) = ||X_i - X_j||_p$. The smallest L is called the Lipschitz constant, which can be denoted as:

$$L_c = \sup_{X_i, X_j \in \mathbb{X}} \frac{|f(X_i) - f(X_j)|}{||X_i - X_j||_p}, \tag{8}$$

L_c represents the maximum ratio between variations in the model's output and variations in its input, which can be used to measure Lipschitz Continuity. Usually, smaller Lipschitz constants represent better Lipschitz continuity and improved generalization. As suggested in Virmaux & Scaman (2018); Weng et al. (2018), the Lipschitz constant in Equation 8 can be computed using the gradient norm as in Lemma 1.

Lemma 1. *If a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^1$ is Lipschitz continuous on $\mathbb{X} \subset \mathbb{R}^m$, the Lipschitz constant in Eq.8 can be calculated by:*

$$L_c = \max_{X \in \mathbb{X}} ||\nabla f(X)||_q, s.t. \frac{1}{p} + \frac{1}{q} = 1, \tag{9}$$

where $\nabla f(X) = (\frac{\partial f(X)}{\partial x_1}, \frac{\partial f(X)}{\partial x_2}, ..., \frac{\partial f(X)}{\partial x_m})$.

A common setting is that $p = q = 2$. In this paper, we calculate L_c of the aggregated models on the test set. The inputs of the aggregated model are the pixels of an image and the output is a vector $Y = (y_1, y_2, ..., y_n)$ representing class probabilities, where n is the number of classes. Note that there are n functions in the model as each element in Y is the output of an independent function. We only calculate L_c of the function corresponding to its real label for each test data.

D LIMITATIONS AND BROADER IMPACTS

Limitations. In this paper, we propose FedBiF to improve bidirectional communication efficiency for federated learning. Extensive experiments have proved that FedBiF enjoys much better communication efficiency and test accuracy than recent state-of-the-art methods. However, we point out that FedBiF has certain limitations. Since a weight in local models is replaced by some virtual bits, the local models may take more memory and a little extra computation to restore the weight. Nonetheless, the extra memory and computation can also be mitigated to some extent by alternately activating each bit, as FedBiF does by default. Specifically, when activating one bit in each round, we can calculate the sum of the frozen bits in advance and add the value of the activated bit in the forward pass. In this way, FedBiF incurs one more virtual bit store and one extra summation for each weight. Furthermore, virtual bits determine the value of their binary bits through signs and do not require exact representations. Thus, we believe that setting virtual bits in FP16 or FP8 format can also satisfy numerical precision and further reduce memory usage.

Broader Impacts. Another challenge of federated learning is the data privacy. Generally, locally trained models can be stolen when uploaded to the server, resulting in leakage of local data information. FedBiF can enhance the privacy protection ability of federated learning to a certain extent, by uploading only several activated bits. Even if the activated bits uploaded by clients are stolen, the parameters or the gradients of the local model remain unknown to the attackers.