APPENDIX In this appendix we introduce the datasets used in the experiments and clarify our data partition method in Appendix A. Network architectures are illustrated in Appendix B. The grid search of baselines' hyperparameters are detailed in Appendix C. Model statistics on different datasets collected from `pFedVEM` are presented in Appendix D. More experimental results are given in Appendix E.

A. Datasets

In this section we introduce the datasets and data partition method used in this paper. We use Fashion-MNIST (FMNIST) and CIFAR10 to model *label distribution skew*, CIFAR100 and SUN397 to model *label concept drift*.

FMNIST [35]. This is a dataset of clothing images consisting of 60,000 training data points and 10,000 test data points associated with 10 labels: [T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot].

CIFAR10 [15]. This dataset consists of 50,000 training data points and 10,000 test data points associated with 10 labels: [Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck].

CIFAR100 [15]. This dataset consists of 50,000 training data points and 10,000 test data points associated with 100 labels. The 100 labels (subclasses) are grouped into 20 superclasses such that every superclass contains 5 subclasses, e.g. the superclass Household funiture contains [Bed, Chair, Couch, Table, Wardrobe].

SUN397 [36]. This scene category dataset contains 108,753 images from 397 categories. It is arranged in a 3-level tree, we use the first level as superclasses: [Indoor, Manmade outdoor, Nature outdoor] and leaf nodes as subclasses, e.g. Indoor contains [Living room, Bedroom, Kitchen, Bathroom, ...]. The number of data points per category and the number of categories per superclass is inconsistent, we take the first 50 categories per superclass and first 100 data points per category according to the ordering in the hierarchy file, then split 80% and 20% per category for the training and test set.

Most existing works conduct splitting to generate clients' data. The splitting methods are either subject to a fixed size or a random distribution. The former (e.g. [1,40]) cannot represent *data quantity disparity*. While the latter methods draw a sequence of fractions per label w.r.t. a Dirichlet distribution $Dir_J(\alpha)$ (e.g. [7, 37]) or a Uniform distribution of a range around 0.5 [29] then split and distribute one fraction to each client. As a result the expected local data sizes are still the same across clients, and *data quantity disparity* cannot be well represented. Some work (e.g. [31]) resorts to sampling, but to avoid the out-of-index problem the sampling range is usually conservative.

In this work we conduct splitting by random slicing, the detailed processing steps are as follows:

(i) Sampling labels or subclasses: First we determine what types of data every client contains. Generating a list of labels (for *label distribution skew*) or lists of subclasses (for *label concept drift*), each client samples without replacement from the list of labels or lists of subclasses. When the list is empty, refill and continue until the for-loop over the clients is done.

(ii) Sampling data points: Then we determine the exact data points every client receives. Assume the type of data needs to be partitioned into M parts for M clients, while there are N data points belonging to this type. We first shuffle the data point vector and then draw $M-1$ indices from 1 to $N-1$ to slice the N data points into M parts. Finally, we distribute the m-th part to the m-th client allocating this type of data.

There are two strengths of such data splitting. First, the federated group always contains the full training set, while we can make data scatter in different patterns via random seeds. Second, the resulting data partition is close to the negative binomial distribution with one success and thus subject to the nature of *data quantity disparity*, that is, few big datasets are concentrated on few clients, whereas a large amount of data is scattered acros many clients with small dataset sizes. The intuition behind is that when we slice the data points vector at step (ii), $M-1$ indices are uniformly draw from $1, \ldots, N-1$. Thus every index approximately conducts a Bernoulli trial with $p = (M-1)/(N-1)$, although in fact they are dependent. We visualize the distribution of local data sizes in Figure 3 - Figure 6. We notice

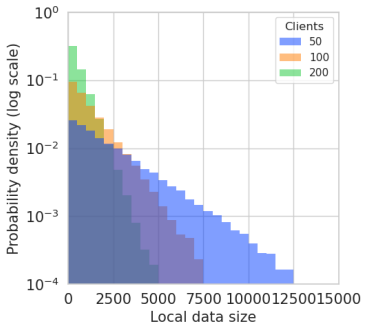


Figure 3. Distribution of local data sizes on FMNIST (setting is consistent with Table 1). Visualized by 1000 times Monte-Carlo simulation.

that the distribution on SUN397 is different from others. Recall in our setting, SUN397 has 50 subclasses per sub-

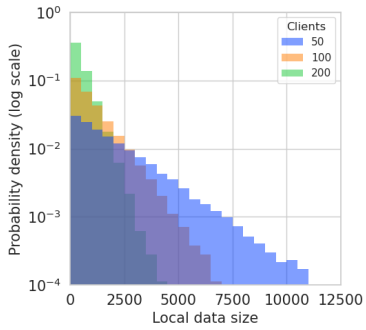


Figure 4. Distribution of local data sizes on CIFAR10 (setting is consistent with Table 1). Visualized by 1000 times Monte-Carlo simulation.

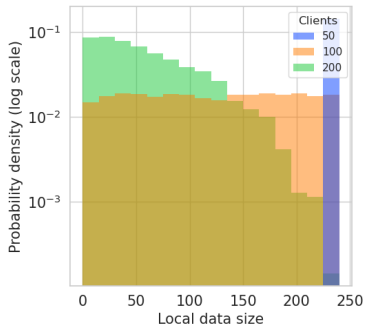


Figure 5. Distribution of local data sizes on SUN397 (setting is consistent with Table 1). Visualized by 1000 times Monte-Carlo simulation.

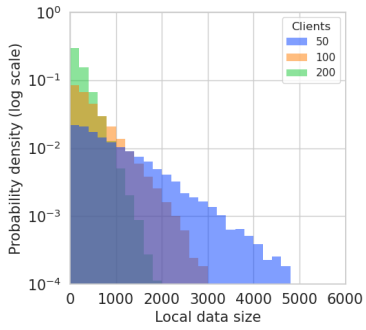


Figure 6. Distribution of local data sizes on CIFAR100 (setting is consistent with Table 1). Visualized by 1000 times Monte-Carlo simulation.

class. When #Clients $= 50$ every subclass in SUN397 is distributed to exactly one client and when #Clients $= 100$ every subclass is distributed to two clients, thus in these two extreme cases the resulting PDF is either a delta or a uniform function.

B. Network Architectures

In this section we illustrate the network architectures used in this work, see Figure 7 - Figure 9.

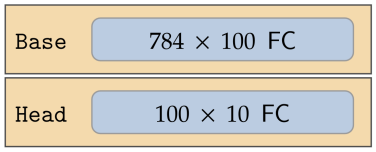


Figure 7. Network architecture for FMNIST.

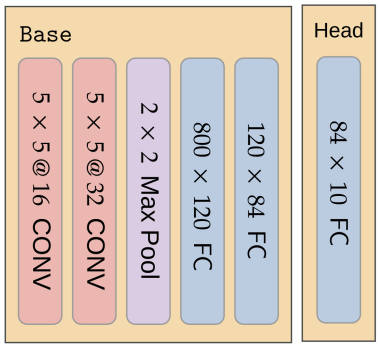


Figure 8. Network architecture for CIFAR10.

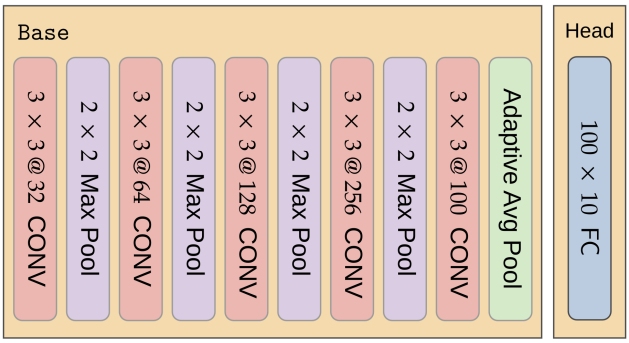


Figure 9. Network architecture for CIFAR100 and SUN397.

C. Hyperparameters

In this section we describe the hyperparameter search space for each baseline and grid search is used. The search space is determined according to the best hyperparameters provided in previous works. The notations of hyperparameters for each framework are detailed below, the search space is summarized in Table 4.

`FedAvg`: We tune on the learning rate η, batch-size B and epochs R.

`FedProx`: We tune on the learning rate η, batch-size B, epochs R, and the penalty constant μ in the proximal term.

`FedPer`: We tune on the learning rate η, batch-size B and epochs R.

`Scaffold`: We adopt Option 2 and set global learning rate to 1, which are suggested by [14]. We tune on the local learning rate η, batch-size B and epochs R.

`FedRep`: We tune on the learning rate η, batch-size B, epochs R for the base model and epochs K for the head model.

`PerFedAvg`: We apply the Hessian-free (HF) variant which outperforms the first-order (FO) variant in most settings. We tune on the stepsize α for the adaptation, and the learning rate β, iterations R, batch-size B for the local training of the Meta-model. Before evaluation, every client adapts the Meta-model to the local data with one epoch training using batch-size B, which empirically performs better than one-step adaptation.

`IFCA`: We follow the configuration for ambiguous cluster structure and define the number of clusters to two, which are suggested by [9]. We tune on the learning rate η, batch-size B and epochs R.

`pFedME`: Following [31], the global model update factor β is set to be 1, which is more stable when the number of clients is changed or different random seed is used. We tune on the learning rate η, batch-size B, regularization constant λ and local computation rounds R.

`pFedBayes`: Like [31], the global model update factor β is set to be 1. Same as `pFedVEM`, the batch-size is set to be the local data size. We tune on the learning rate η, epochs R, initial standard deviation σ and regularization constant λ.

`Local`: Different from FL frameworks which apply the same hyperparameters across clients, for `Local` we allow every client to train a model locally for 20 epochs while search for the respective best batch-size B and learning rate η.

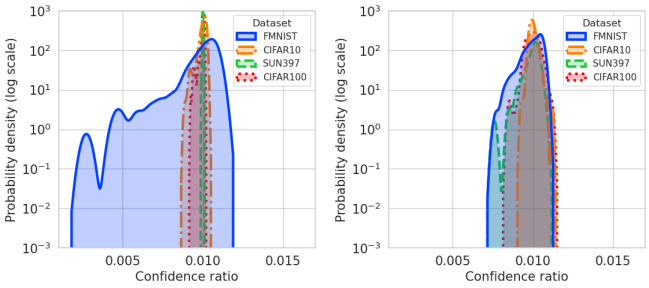
Method	Hyperparameter	Search Range
FedAvg	η	{0.01, 0.001, 0.0001}
	R	{5, 10, 20}
	B	{10, 50, 100}
FedProx	η	{0.01, 0.001, 0.0001}
	R	{5, 10, 20}
	B	{10, 50, 100}
	μ	{0.1, 1, 10}
Scaffold	η	{0.1, 0.01, 0.001}
	R	{5, 10, 20}
	B	{10, 50, 100}
FedPer	η	{0.01, 0.001, 0.0001}
	R	{5, 10, 20}
	B	{10, 50, 100}
FedRep	η	{0.01, 0.001, 0.0001}
	R	{5, 10, 20}
	K	{5, 10, 20}
	B	{10, 50, 100}
PerFedAvg	α	{0.1, 0.01, 0.001}
	β	{0.1, 0.01, 0.001}
	R	{5, 25, 50}
	B	{10, 50, 100}
IFCA	η	{0.01, 0.001, 0.0001}
	R	{5, 10, 20}
	B	{10, 50, 100}
pFedME	η	{0.01, 0.001, 0.0001}
	λ	{1, 10, 15}
	R	{5, 10, 20}
	B	{10, 50, 100}
pFedBayes	η	{0.01, 0.001, 0.0001}
	λ	{1, 10, 15}
	R	{5, 10, 20}
	σ	{1, 0.1, 0.01, 0.001}
Local	η	{0.001, 0.0001}
	B	{10, 50, 100}

Table 4. Hyperparameters and the corresponding search space of the baselines.

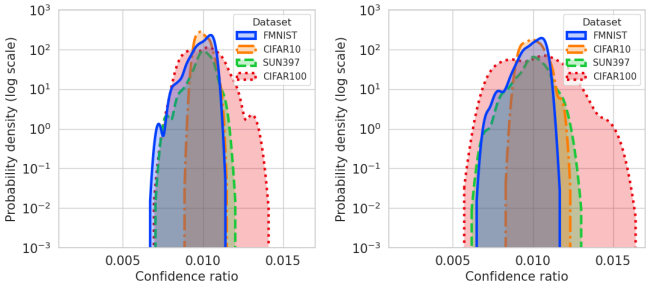
D. Model Statistics

In this section we discuss the model statistics measured by `pFedVEM`. We first investigate the distribution of confidence values in different settings. Since in the model aggregation, the parameters of client j is weighted by $\tau_j / \sum_{j \in \mathcal{S}_t} \tau_j$ (cf. Equation (9)), we thus investigate the confidence ratio $\tau_j / \sum_{j=1}^{J} \tau_j$ instead. We collect clients' confidence ratios at different communication rounds from 5 independent runs and visualize the distribution using kernel density estimation with Gaussian kernel. Figure 10 shows that at the end of training, clients trained on CIFAR100 exhibit the largest variation of confidence ratio.

We then investigate the distribution of model deviation. For the j-th client the model deviation is defined as $\|\boldsymbol{w}_j - \boldsymbol{w}\|^2/d$. Similarly, we summarize the results of 5 runs and visualize the distribution using kernel density estima-

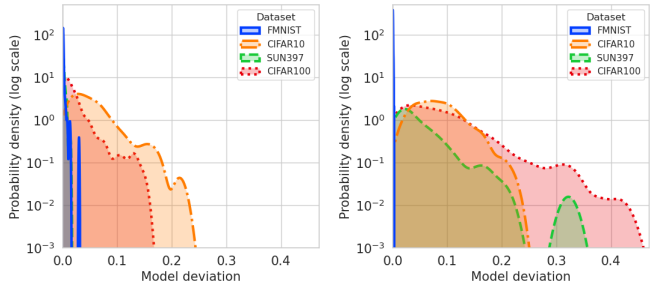


(a) Communication round = 5 (b) Communication round = 30

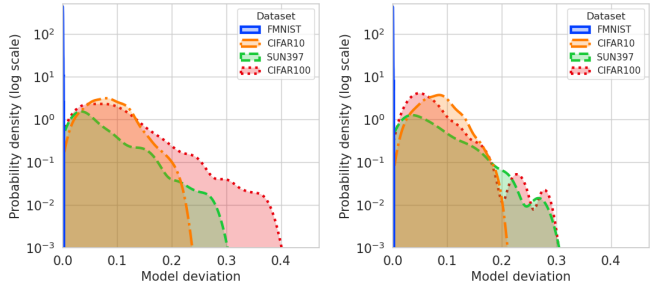


(c) Communication round = 60 (d) Communication round = 100

Figure 10. Distribution of $\tau_{1:J}/\sum_{j=1}^{J}\tau_j$ over 100 clients on different datasets. Kernel density estimation is used for the visualization.



(a) Communication round = 5 (b) Communication round = 30



(c) Communication round = 60 (d) Communication round = 100

Figure 11. Distribution of $\|\boldsymbol{w}_{1:J}-\boldsymbol{w}\|^2/d$ over 100 clients on different datasets. Kernel density estimation is used for the visualization.

tion. Figure 11 shows that during training, clients models trained on CIFAR100 spreads out over a larger range. This indicates that the setting of CIFAR100 is highly heterogeneous and this task is a more challenging as the clients' parameters sent back to the server could be severely deviated from each other. In contrast, clients trained on FMNIST concentrates on the global model.

E. Additional Experimental Results

E.1. Convergence rate

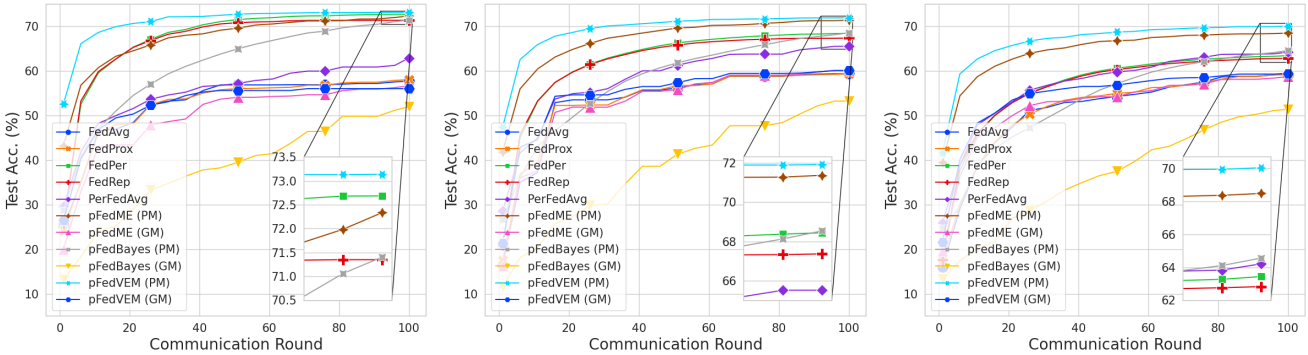
Since the communication bottleneck is one of the main issues of FL, a framework with faster convergence rate is preferred. We therefore present the test accuracy vs. communication round plots evaluated on CIFAR10 over 50, 100, 200 clients. As shown in Figure 12, `pFedVEM` converges faster than other frameworks and already reaches a good accuracy at 30 rounds of communication. This strength of `pFedVEM` is more obvious for 50 clients, where each client is expected to have more data and therefore variational inference performs better.

E.2. Limited Data Availability

We examine two cases to demonstrate the advantage of our method `pFedVEM` when data is scarce. First, we show that the small clients, i.e. the 10% of clients with the smallest local data sizes, perform better with `pFedVEM`. Based on Table 5, we see the utility gap between the small clients and the overall average is moderate for `pFedBayes` and our method, while our method `pFedVEM` achieves significantly better accuracy than the baselines. Second, we reduce the number of training samples $|\mathcal{D}|$ that the federated group of all clients has and evaluate the performance of all

Dataset	Method	Small Clients	All Clients	Diff.
CIFAR10	FedPer	66.2 ± 0.0	68.4 ± 0.4	-2.2
	FedRep	64.5 ± 0.0	67.4 ± 0.4	-2.9
	PerFedavg	51.8 ± 0.0	65.6 ± 0.8	-13.8
	pFedME	65.0 ± 0.0	71.4 ± 0.2	-6.4
	pFedBayes	66.9 ± 0.0	68.5 ± 0.3	-1.6
	Ours	$\mathbf{70.2 \pm 0.0}$	$\mathbf{71.9 \pm 0.1}$	-1.7
CIFAR100	FedPer	30.1 ± 0.0	39.3 ± 0.7	-9.2
	FedRep	30.8 ± 0.0	41.2 ± 0.6	-10.4
	PerFedavg	45.2 ± 0.0	48.3 ± 0.5	-3.1
	pFedME	40.5 ± 0.2	47.6 ± 0.5	-7.1
	pFedBayes	40.6 ± 0.0	46.5 ± 0.2	-5.9
	Ours	$\mathbf{50.8 \pm 0.2}$	$\mathbf{56.2 \pm 0.4}$	-5.4

Table 5. Average test accuracy of PMs (% $\pm$ SEM) of small clients (10% clients with the smallest local data sizes) and all clients. The difference between the means is shown in the last column. The experimental configuration corresponds to the 100-client setting in Table 1.



(a) Client = 50 (b) Client = 100 (c) Client = 200

Figure 12. Convergence rate evaluated by test accuracy vs. communication round over 50, 100, 200 clients on CIFAR10.

Dataset	Method	$\|\mathcal{D}\| = 5000$		$\|\mathcal{D}\| = 10000$		$\|\mathcal{D}\| = 50000$	
		PM	GM	PM	GM	PM	GM
CIFAR10	Local	35.7 ± 0.3	–	39.6 ± 0.1	–	87.5 ± 0.1	–
	FedAvg	–	41.2 ± 0.6	–	47.4 ± 1.0	–	85.4 ± 0.3
	FedProx	–	41.5 ± 0.5	–	46.6 ± 0.5	–	$\mathbf{86.3 \pm 0.2}$
	Scaffold	–	39.7 ± 0.6	–	46.4 ± 0.3	–	85.4 ± 0.1
	FedPer	40.7 ± 0.5	–	51.8 ± 0.9	–	90.7 ± 0.1	–
	FedRep	40.7 ± 0.5	–	51.3 ± 0.9	–	90.7 ± 0.1	–
	IFCA	41.3 ± 0.6	–	47.0 ± 0.7	–	85.6 ± 0.2	–
	PerFedavg	41.4 ± 0.4	–	48.2 ± 0.3	–	88.6 ± 0.1	–
	pFedME	44.9 ± 0.9	36.1 ± 1.1	53.1 ± 1.2	42.5 ± 1.8	$\mathbf{91.4 \pm 0.1}$	84.4 ± 0.6
	pFedBayes	52.3 ± 0.5	36.8 ± 0.4	58.1 ± 0.3	41.9 ± 0.9	91.3 ± 0.1	84.2 ± 0.3
	Ours	$\mathbf{57.1 \pm 0.2}$	$\mathbf{45.7 \pm 0.3}$	$\mathbf{61.8 \pm 0.3}$	$\mathbf{50.1 \pm 0.4}$	$\mathbf{91.4 \pm 0.1}$	85.6 ± 0.2

Table 6. Average test accuracy of PMs and test accuracy of GM (% $\pm$ SEM) over different numbers of training samples of CIFAR10. Other configurations correspond to the 100-client setting in Table 1. Best result is in bold.

Dataset	Method	50 Clients		100 Clients		200 Clients	
		PM	GM	PM	GM	PM	GM
Digit-Five	FedProx	–	86.2 ± 0.4	–	86.1 ± 0.1	–	85.8 ± 0.3
	pFedME	91.2 ± 0.2	85.6 ± 0.4	89.5 ± 0.3	86.6 ± 0.3	88.4 ± 0.1	$\mathbf{86.9 \pm 0.6}$
	pFedBayes	90.9 ± 0.2	74.8 ± 1.3	88.2 ± 0.2	76.1 ± 0.6	85.2 ± 0.3	74.4 ± 0.7
	Ours	$\mathbf{92.7 \pm 0.2}$	$\mathbf{86.6 \pm 0.3}$	$\mathbf{91.1 \pm 0.1}$	$\mathbf{87.1 \pm 0.1}$	$\mathbf{89.8 \pm 0.1}$	$\mathbf{86.9 \pm 0.2}$

Table 7. Average test accuracy of PMs and test accuracy of GM (% $\pm$ SEM) over 50, 100, 200 clients on Digit-Five. Best result is in bold.

frameworks. Again, results in Table 6 show that our method performs significantly better than the baselines when data is scarce.

E.3. Feature Distribution Skew

We study a mixed case of *label distribution skew* and *label concept drift*, which is also known as *feature distribution skew*. To this end, we use the Digit-Five dataset consisting of MNIST, SVHN [25], USPS [12], MNIST-M [8], Synthetic Digits [28] and adopt the following allocation scheme: 1) randomly select a dataset 2) randomly pick 5 labels in the selected dataset 3) randomly select data and distribute to each client. We tune hyperparameters of three main competing baselines. The results are presented in the Table 7. We note that all methods on Digit-Five obtain overall high accuracy, while our method outperforms other methods in this setting of feature distribution skew.