APPENDIX

A EXPERIMENTAL SETUP FOR SECTION 3

Federated Learning Setup. In *Sec. 3 Limitations of Previous Work and Insights*, we employ CIFAR10 for an empirical comparison. We conduct six groups of experiments to simulate different levels of label skew and scarcity, as introduced below.

- **(a)** The training set of CIFAR10 is uniformly distributed to 10 clients, resulting in each local dataset having a size of 5000 ($|\mathcal{D}_k| = 5000$) and 10 classes (IID). More specifically, each local dataset has 500 samples per class.
- **(b)** CIFAR10 is uniformly distributed to 100 clients. Thus each local dataset has a size of 500 ($|\mathcal{D}_k| = 500$), 10 classes (IID) with each class containing 50 samples.
- **(c)** CIFAR10 is uniformly distributed to 500 clients. Each local dataset have a size of 100 ($|\mathcal{D}_k| = 100$), 10 classes (IID) with each class containing 10 samples.
- **(d)** The training set of CIFAR10 is distributed to 10 clients, but only three out of ten classes are distributed to each client (non-IID), resulting in each local dataset having a size of 5000 ($|\mathcal{D}_k| = 5000$) and approximately 1666 samples for each class.
- **(e)** Similar to (d), CIFAR10 is distributed to 100 clients ($|\mathcal{D}_k| = 500$) with each client containing 3 classes (non-IID). This data distribution is shown in Figure 9(a).
- **(f)** Similar to (d), CIFAR10 is distributed to 500 clients ($|\mathcal{D}_k| = 100$). Each client contains 3 classes (non-IID) with about 33 samples per class.

For CIFAR10 classification, we employ MobileNet_V2, which has 18 blocks consisting of multiple convolutional and pooling layers (Sandler et al., 2018). We use the Adam optimizer for local training with an initial learning rate of 10^{-3} and decay it by 2% per communication round until 10^{-5}. For (a) and (d), all clients will participate in the training in each round, while for the other groups, we will randomly select 10% of the clients for each round. The size of the local batch is 64, and we run 10 local epochs for groups (a, b, d, e) and 5 local epochs for groups (c, f). We run 100 communication rounds for all groups to ensure global convergence.

Experimental setup for Figure 1: To compare the performance of existing methods with , we use CIFAR10 dataset and report the classification accuracy of the global model based on the global testing set. We compare *FedAvg* with loss-based methods such as *FedDecorr* and *FedNTD*, as well as data augmentation-based methods like *FedMix* and *FedData*. They are the most representative methods in each category. *FedMix* is implemented by averaging every 10 samples and sharing the result globally. The shared averaged data is then combined with local data according to a Beta distribution (with the $a = 2$) for local training. In the case of *FedData*, we collect 10% of the data (randomly chosen) from each client and share it globally, in the first communication round. To simulate varying scarcity levels, we split the CIFAR10 training set (comprising $50,000$ samples in total) into 5000, 500, and 100 training samples on average per client, which ends up with 10, 100 and 500 clients finally. Other settings are the same with the main experiments as introduced in Sec. 5.1.

Experimental setup for Figure 2: DB score (Davies & Bouldin, 1979) is defined as the average similarity measuring each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering. To calculate the score for features, we use the ground-true class labels as cluster labels, and use Euclidean distance between features to measure the similarity.

For a fair comparison, the local training for all clients starts from a same global status with an accuracy of 40%. The features of the testing set from the initial global model present a DB of 4.8. We run one communication round and report the performance for the global model. In this round, for $|\mathcal{D}_k| = 5000$ we aggregate 10 clients while for $|\mathcal{D}_k| = 100$ we aggregate 50 clients, so that the total samples used for model training are kept unchanged. For $|\mathcal{D}_k| = 100 + 1000$ group, we additionally give the selected 50 clients 1000 samples (gathered in the first round) to aid local training. In Figure 2, for local models, we report the averaged DB across clients.

B NOTATIONS AND ALGORITHM

Table 3: Notations used in this paper.

$\|\mathcal{C}\|$	The number of classes	T	The number of rounds
$\|K\|$	The number of clients	k	Client k
$\mathcal{D}_k$	client k's data	$\mathcal{D}$	Global data
l	From layer l to extract features	$\theta^{(t)}$	Global model at t-th round
$\theta_{k,:l}$	Top l layers of the model	$\theta_{k,l:}$	Layers after l of the model
$\mathcal{F}_k^{(t)}$	Local feature set	$\mathcal{F}^{(t)}$	Global feature set
a	Parameter for Beta distribution	λ	Weigh for feature mix-up in loss
α	Local feature sampling fraction	$\lambda 2, \lambda 3$	Weigh in loss

Algorithm 1: Federated Learning with Feature Sharing (FLea)

```
Input  : Total rounds $T$, local learning rate $\eta$, local training epochs $E$, sampled clients set $\mathcal{K}^{(t)}$,
         a given layer $l$, parameter $a$ for Beta distribution.
Output: Global model $\theta^{(T)}$.
Initialize $\theta^{(0)}$ for the global model
for each round $t$ = 1,2,...,$T$ do
    Server samples clients $\mathcal{K}^{(t)}$ and broadcasts $\theta_k \leftarrow \theta^{(t-1)}$
    Server broadcasts the feature $\{\mathcal{F}^{(t)}, .., \mathcal{F}^{(t-\tau)}\}$ to clients in $\mathcal{K}^{(t)}$    // Skip if $t = 1$.
    for each client $k \in \mathcal{K}^{(t)}$ in parallel do
        for local step $e = 1, 2, .., E$ do
            for local batch $b = 1, 2, ...$ do
                sample $\lambda \sim Beta(a, a)$
                $\theta_k \leftarrow \theta_k - \eta \nabla \mathcal{L}(\theta_k)$  // if $t = 1$, only use local data for training. Otherwise,
                  use one batch of local data $\mathcal{D}_k$ and one batch of global feature $\mathcal{F}^{(t)}$ according
                  to Eq. (6).
            end
        end
        Client $k$ sends $\theta_k$ to server
    end
    Server aggregates $\theta_k$ to a new global model $\theta^{(t)}$ refer to Eq. (1)
    for each client $k \in \mathcal{K}^{(t)}$ in parallel do
        Client $k$ receives model $\theta^{(t)}$
        Client $k$ extracts (without gradients) and sends $\mathcal{F}_k^{(t)}$ to server
    end
end
```

Beta Distribution. The probability density function (PDF) of the Beta distribution is given by,

$$f(\lambda; a, b) = \frac{\lambda^{a-1}(1-\lambda)^{(b-1)}}{N}, \tag{7}$$

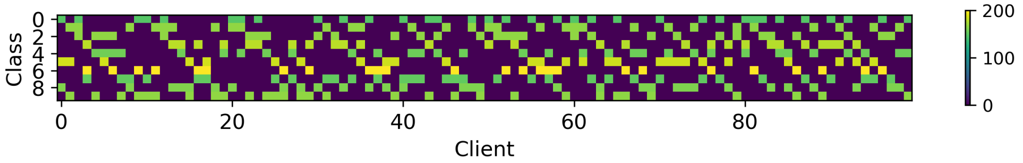
where N is the normalizing factor and $\lambda \in [0, 1]$. In our study, we choose $a = b$ and herein, $f(\lambda) = \frac{1}{N}(\lambda(1-\lambda))^{a-1}$.

C DETAILS OF EXPERIMENTS

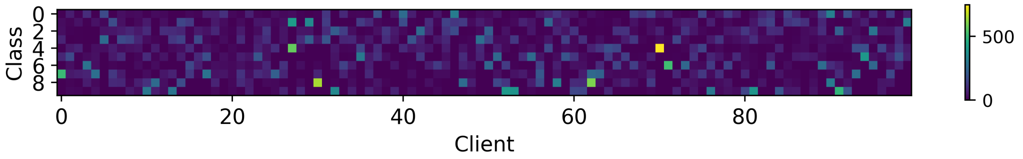
C.1 DATA DISTRIBUTION

Image data: We test our algorithm on CIFAR10 (Krizhevsky et al., 2009). We distribute CIFAR10 training images (containing $50,000$ samples for 10 classes) to $K = 100$ and $K = 500$ clients and use the global CIFAR10 test set (containing $1,000$ samples per class) to report the accuracy of the global model. We show the data splits for 100 clients setting in Figure 9. For 500 clients setting the distribution is similar, but the number of samples per client reduces to one-tenth of the number shown in Figure 9.

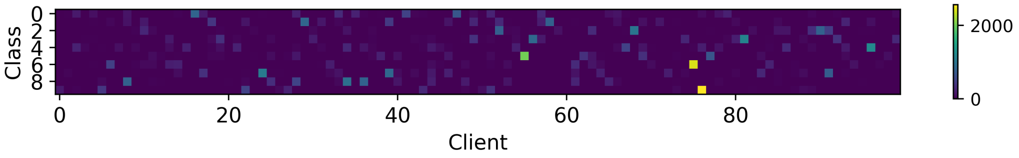
Audio data: We also test *FLea* using UrbanSound8K dataset Salamon et al. (2014). This dataset contains 8732 labeled sound excerpts ($\leq 4s$) of urban sounds from 10 classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. For experiments, we randomly hold out 20% (about 1700 samples) for testing and distribute the rest (about 7000 samples) to K clients for training. We report the results for $K = 70$ and $K = 140$, using the $Quantity(3)$, $Dirichlet(0.5)$, and $Dirichlet(0.1)$ splits.



(a) Quantify(3).

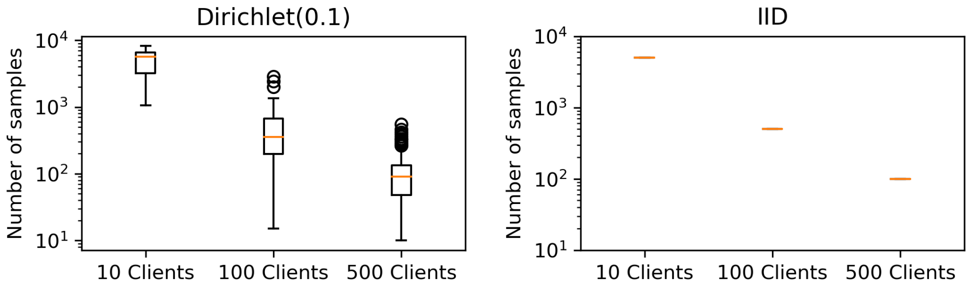


(b) Dirichlet(0.5).



(c) Dirichlet(0.1).

Figure 9: Training data split for CIFAR10, $|K| = 100$.



(a) With label skew. (b) Without label skew.

Figure 10: The distribution of the number of samples per client $|\mathcal{D}_k|$ for CIFAR10.

To better illustrate the data scarcity problem, we visualize the distribution for the local data size in Figure 10. As it shown, when we distribute the training set of CIAFR10 ($50,000$ samples) to 10 clients using a Dirichlet distribution parameterized by 0.1, these clients will present different class distributions, and the total number of local samples ranges from 2853 to 8199. This is the commonly explored non-IID setting. In this paper, we further explore scarce non-IID data, and thus we split the data into 100 and 500 clients. As a result, the number of samples per client reduces significantly: the median number drops from 5685 to 90 when the number of clients increases from 10 to 500, as shown in Figure 10(a). This is the realistic scenario that we are interested in. It is also worth mentioning that data scarcity is independent of label skew and it can happen in the IID scenario. As shown in Figure 10(b), the local data covers 10 classes uniformly, but the data scarcity problem becomes severe when the number of client increase.

C.2 MODEL ARCHITECTURE AND HYPER-PARAMETERS

We classify images in CIFAR10 using MobileNet_V2 (Sandler et al., 2018) that has 18 blocks consisting of multiple convolutional and pooling layers. The architectures of MobileNet_V2 for CIFAR10 is summarized in Table 4.

For audio classification, the audio samples are first transformed into spectrograms and fed into a CNN model, which we termed as *AudioNet*. This model consists of four blocks, each comprising a convolutional layer, a Relu activation layer, and a Batch Normalization layer, followed by a fully connected layer[1]. The details of the convolutional layers are summarized in Table 5.

We use the Adam optimizer for local training with an initial learning rate of 10^{-3} and decay it by 2% per communication round until 10^{-5}. The size of the local batch is 64, and we run 10 local epochs for 100 clients setting and 15 local epochs for the rest. For feature augmentation, we use $Beta(2, 2)$. The weights in the loss function are set to $\lambda_1 = 1$ and $\lambda_2 = 3$. 10% of clients are randomly sampled at each round. We run 100 communications and take the best accuracy as the final result. For all results, we report the mean and standard deviation of the accuracy from five runs with different random seeds.

Table 4: Architecture of MobileNet_V2. Features used to report the results in Table 6 are underlined.

Block(CNN layers)	#Input	Operator	#Output Channel	#Kernel	#Stride	#Output
$0(1)$	$3 \times 32 \times 32$(image)	conv2d	32	3	1	$32 \times 32 \times 32$
$1(2-5)$	$32 \times 32 \times 32$	conv2d×4	$32, 32, 16, 16$	$1, 3, 1, 1,$	$1, 1, 1, 1$	$\underline{16 \times 32 \times 32}$
$2(6-9)$	$16 \times 32 \times 32$	conv2d×4	$96, 96, 24, 24$	$1, 3, 1, 1$	$1, 1, 1, 1$	$32 \times 32 \times 32$
$3(10-12)$	$32 \times 32 \times 32$	conv2d×3	$144, 144, 24$	$1, 3, 1$	$1, 1, 1$	$24 \times 32 \times 32$
$4(13-14)$	$24 \times 32 \times 32$	conv2d×3	$144, 144, 32$	$1, 3, 1$	$1, 2, 1$	$32 \times 16 \times 16$
$5\&6(15-20)$	$32 \times 16 \times 16$	conv2d×3	$192, 192, 32$	$1, 3, 1$	$1, 1, 1$	$32 \times 16 \times 16$
$7(21-23)$	$32 \times 16 \times 16$	conv2d×3	$192, 192, 64$	$1, 3, 1$	$1, 2, 1$	$64 \times 8 \times 8$
$8, 9, \&10(24-32)$	$64 \times 8 \times 8$	conv2d×3	$384, 384, 64$	$1, 3, 1$	$1, 1, 1$	$64 \times 8 \times 8$
$11(33-36)$	$64 \times 8 \times 8$	conv2d×4	$384, 384, 96, 96$	$1, 3, , 11$	$1, 1, 1, 1$	$9 \times 8 \times 8$
$12\&13(37-42)$	$96 \times 8 \times 8$	conv2d×3	$576, 576, 96$	$1, 3, 1$	$1, 1, 1$	$96 \times 8 \times 8$
$14(43-45)$	$96 \times 8 \times 8$	conv2d×3	$576, 576, 160$	$1, 3, 1$	$1, 2, 1$	$160 \times 4 \times 4$
$15\&16(46-51)$	$160 \times 4 \times 4$	conv2d×3	$960, 960, 160$	$1, 3, 1$	$1, 1, 1$	$160 \times 4 \times 4$
$17(52-54)$	$160 \times 4 \times 4$	conv2d×3	$960, 960, 320$	$1, 3, 1$	$1, 1, 1$	$320 \times 4 \times 4$
$18(55)$	$320 \times 4 \times 4$	conv2d	1280	1	1	$1280 \times 4 \times 4$

Table 5: Architecture of AduioNet. Features used to report the results in Table 7 are underlined.

Index	#Input	Operator	#Output Channel	#Kernel	#Stride	#Output
1	$2 \times 64 \times 344$(2-channel spectrogram)	conv2d	8	5	2	$8 \times 32 \times 172$
2	$8 \times 32 \times 172$	conv2d	16	3	2	$\underline{16 \times 16 \times 86}$
3	$16 \times 16 \times 86$	conv2d	32	3	2	$32 \times 8 \times 43$
4	$32 \times 8 \times 43$	conv2d	64	3	2	$64 \times 4 \times 22$

C.3 BASELINE IMPLEMENTATION

More details for baseline implementations are summarized as blew,

- ***FedProx***: We adapt the implementation from (Li et al., 2020b). We test the weight for local model regularization in $[0.1, 0.01, 0.001]$ and report the best results.
- ***FedLC***: it calibrates the logits before softmax cross-entropy according to the probability of occurrence of each class (Zhang et al., 2022a). We test the scaling factor in the calibration from 0.1 to 1 and report the best performance.

---

[1] `https://www.kaggle.com/code/longx99/sound-classification/notebook`

Table 6: Overall performance comparison. Accuracy is reported as $mean \pm std$ across five runs. The best baseline (excluding *FedData*) under each column is highlighted.

	#Clients: 100 (500 samples per client on average)			**#Clients: 500 (100 samples per client on average)**		
	$Quantity(3)$	$Dirichlet(0.5)$	$Dirichlet(0.1)$	$Quantity(3)$	$Dirichlet(0.5)$	$Dirichlet(0.1)$
FedAvg	43.55±0.82	50.36±0.89	28.21±1.20	30.25±1.33	32.58±1.09	20.46±2.15
FedProx	44.37±0.89	49.30±1.00	34.66±1.11	31.92±1.45	32.01±1.25	20.86±1.97
FedDecorr	44.09±0.90	51.27±0.93	30.89±1.40	31.12±1.57	33.57±1.22	21.34±1.59
FedLC	49.35±1.01	53.58±1.02	36.05±1.21	32.05±1.60	30.17±1.18	18.82±2.01
FedNTD	53.01±1.23	56.06±0.97	41.48±0.90	39.98±0.97	39.82±0.86	26.78±2.34
FedBR	44.58±0.73	51.65±1.02	32.11±1.45	31.66±1.07	33.08±1.12	20.98±2.54
CCVR	49.11±0.67	51.21±0.98	34.47±1.35	35.95±1.63	35.02±1.43	24.21±2.67
FedGen	46.66±2.87	52.89±1.09	33.18±1.29	32.32±1.21	34.27±1.56	22.56±2.89
Each client shares 10% of the data or features						
FedData	67.60±1.33	72.17±1.34	70.34±1.68	54.64±1.02	56.47±1.22	55.35±1.46
FedMix	52.78±1.99	57.97±1.24	40.68±1.50	44.04±1.53	45.50±1.88	38.13±2.06
FLea ($l = 5$)	58.27±0.95	59.63±1.28	43.65±1.47	47.03±1.01	48.86±1.43	44.40±1.23

Table 7: Overall performance comparison for audio classification. Accuracy is reported as $mean \pm std$ across five runs. The best baseline (excluding *FedData*) under each column is highlighted.

	#Clients: 70 (100 samples per client on average)			**#Clients: 140 (50 samples per client on average)**		
	$Quantity(3)$	$Dirichlet(0.5)$	$Dirichlet(0.1)$	$Quantity(3)$	$Dirichlet(0.5)$	$Dirichlet(0.1)$
FedAvg	43.69±0.56	46.77±0.87	34.59±2.64	39.35±0.60	43.98±0.89	31.21±1.62
FedProx	38.45±0.48	39.58±1.02	34.81±0.46	39.05±0.56	42.21±0.76	32.85±1.22
FedDecorr	45.01±0.57	46.77±0.65	35.87±1.03	39.67±0.58	44.23±0.95	33.67±1.34
FedLC	50.98±0.49	50.11±0.83	37.05±0.87	44.33±0.79	45.15±0.80	39.87±1.04
FedNTD	44.80±0.45	51.09±0.97	36.53±0.99	42.21±0.63	48.63±0.78	40.15±1.22
FedBR	44.05±0.63	47.58±0.90	36.15±1.17	41.15±0.70	44.37±0.82	34.89±1.36
CCVR	47.12±0.72	49.26±0.92	39.62±1.20	44.05±0.87	46.68±0.83	36.80±1.37
FedGen	45.20±0.89	48.33±1.12	38.27±1.44	40.89±0.72	44.54±0.81	35.78±1.40
Each client shares 10% of the data or features						
FedData	62.83±1.25	64.45±0.76	61.11±0.98	60.31±0.82	60.48±0.91	59.67±1.55
FedMix	51.56±0.59	54.18±0.62	43.35±0.72	46.55±0.81	50.00±0.92	42.27±1.15
FLea ($l = 2$)	57.73±0.51	59.22±0.78	45.94±0.77	54.35±0.80	55.68±0.87	45.05±1.32

- **FedDecorr**: This method applies a regularization term during local training that encourages different dimensions of the low-dimensional features to be uncorrelated (Shi et al., 2022). We adapt the official implementation[2] and suggested hyper-parameter in the source paper. We found that this method can only outperform *FedAvg* with fewer than 10 client for CIFAR10.
- ***FedNTD***: It prevents the local model drift by distilling knowledge from the global model (Lee et al., 2022). We use the default distilling weights from the original paper as the setting are similar[3].
- **FedBR** (Guo et al., 2023): this approach leverage 32 mini-batch data averages without class labels as data augmentation. A min-max algorithm is designed, where the max step aims to make local features for all classes more distinguishable while the min step enforces the local classifier to predict uniform probabilities for the global data averages. We adapt the official implementation[4] in our framework.
- ***CCVR***: It collects a global feature set before the final fully connected linear of the converged global model, i.e., the model trained via *FedAvg*, to calibrate the classifier on the server (Luo et al., 2021). For a fair comparison, we use the same amount of features as our method for this baseline, and we fit the model using the features instead of distributions as used in (Luo et al., 2021). This allows us to report the optimal performance of *CCVR*.
- **FedGen**: It is a method that trains a data generator using the global model as the discriminator to create synthetic data for local training (Liu et al., 2022). The generator outputs $\hat{x}_i$ with input (y_i, z_i) where z_i is a sample for Normal distribution. The generator is a convolutional neural

---

[2] `https://github.com/bytedance/FedDecorr`

[3] `https://github.com/Lee-Gihun/FedNTD.git`

[4] `https://github.com/lins-lab/fedbr`

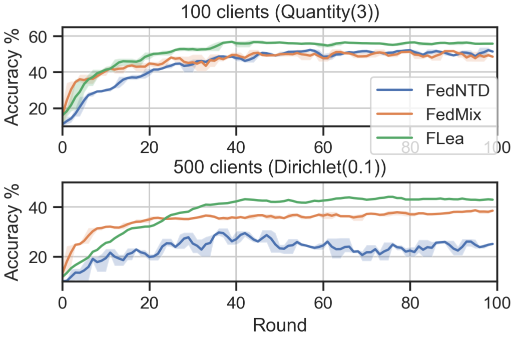


Figure 11: Accuracy of the global model for CIFAR10.

network consisting of four *ConvTranspose2d* layers to upsample feature maps. We train the first 30 rounds by normal *FedAvg* and after 30 rounds, we use the global model as the discriminator to distinguish with the generated data $\hat{x}_i$ is real or not.

- **FedData**: In this baseline, we assume the server waits until all the clients have shared 10% of their local data in the beginning round. The gathered data will be sent to clients to mix with local data for model training.
- **FedMix**: Similar with *FedData*, we assume the server waits until all the clients have shared their data averages.we use a mini-batch of 10 to aggregate the samples. Different from *FedBR* The gathered data will be sent to clients, combing with local data based on the Beta distribution.

C.4 EXTENDED RESULTS

Tables 6 and 7 for additional results corresponding to different local data sizes, supplementing the information presented in Table 2.

Results for CIFAR10 classification. The analysis for Table 6 is elaborated on as follows. Among loss-based approaches, *FedNTD* is the best, showing a strong ability to handle data scarcity and label skew (about 10% improvement from *FedAvg*), while other loss-based FL methods present marginal gain from *FedAvg*. The outstanding performance of *FedNTD* are mainly attributed to its knowledge distillation component, mitigating the local over-fitting as well as model bias. *FLea* further improves *FedNTD* by $3 \sim 5\%$ when the average local data size is 500, and the superiority of *FLea* is more remarkable with increasing level of data scarcity, e.g., when the average local data size reduces to 100, the performance gain reaches 17.6% (*Dirichlet(0.1)* group).

For data augmentation-based baselines, *FedMix* performs the best and for most of the cases, it is the SOTA baseline excluding *FedData*. When sharing the same proportion of global proxies (*FedMix* shares data averages while *FLea* shares features), *FLea* outperforms *FedMix* by $2 \sim 6\%$ across all experiments. We report the performance of *FedData* as an Oracle. It is plausible that *FLea* cannot beat *FedData* given *FedData* shares raw data with privacy protection.

FLea also presents more stable performance compared to *FedNTD* and *FedMix*. As shown in Figure 11, *FLea* converges after 40 communication rounds, with notably higher averaged accuracy and smaller variance compared to the other two best baselines. We also demonstrate each component in *FLea* yields independent contribution to the overall performance in Appendix C Sec. C.5.

Results for UrbanSound8K classification. Similarly to the performance for audio classification, *FLea* consistently achieve the best accuracy across different settings. Given that the total size of UrbanSound8K is smaller than CIFAR10, this audio classification has more sever data scarcity problem globally and locally. This explains why *FedMix* is the best baseline uniformly for this task. Nevertheless, *FLea* outperforms *FedMix* by $2.59\% \sim 7.80\%$.

Table 8: More results for FLea ($\alpha = 10\%$, $|K| = 100$).

	$Quantity(3)$	$Dirichlet(0.5)$	$Dirichlet(0.1)$
FedMix	44.04±1.53	45.50±1.88	38.13±2.06
FLea ($l = 5$)	47.03±1.01	48.86±1.43	44.40±1.23
FLea ($l = 5$, $\lambda = 0.5$)	45.87±1.23	46.91±1.22	42.01±1.14
FLea ($l = 5$, $\lambda_1 = 0$)	45.16±1.06	46.89±1.42	40.98±1.09
FLea ($l = 1$)	49.67±1.12	50.23±1.35	46.17±1.30
FLea ($l = 9$)	44.05±1.11	44.87±1.56	40.19±1.27

Table 9: Supplemental results of CCVR ($\alpha = 10\%$, $|K| = 100$), where global features come from the later layer in different blocks of MobileNet_V2.

	Block18	Block17	Block13	Block9	Block5	Block1	Raw	Our *FLea*
Quantity(3)	49.11±0.67	50.96±0.75	51.24±0.68	52.18±0.89	51.77±0.73	51.50±0.78	51.53±0.81	58.27±0.95
Dirichlet(0.5)	51.21±0.98	51.98±1.03	52.78±1.15	53.04±1.12	53.25±0.99	53.14±1.07	52.85±0.99	59.63±1.28

C.5 ABLATION STUDY FOR FLEA

As introduced in Sec. 4, our *FLea* leverages feature augmentation by combing local and global features according to the weights following a Beta distribution. Now we give the results to demonstrate the advantage of introducing randomness to improve the model generalization: we use fixed λ instead of sampling it from $Beta(2, 2)$ (refer to FLea ($l = 5$, $\lambda = 0.5$) group in Table 8). We also give the results when removing $\mathcal{L}_{dis}$ from the training loss (refer to FLea ($l = 5$, $\lambda_1 = 0$) group in Table 8). It is evident that our complete version of *FLea* always performs the best.

We also discuss the impact of layer l, from which layer the features are extracted. It is a trade-off between between privacy protection and the utilization of features. A smaller l indicates the features are closer to the raw data while the privacy vulnerability increases. In Sec. 5.3, we have demonstrated that $l = 5$ with the de-correlation loss can well defend against privacy attacks in our simulations. In Table 8, we also show that sharing features from $l = 1$ can enhance *FLea* while from $l = 9$ can lead to a slight performance decline. For real-world applications (beyond CIFAR10), we choose l according to the specific performance and privacy requirements. It is also worth mentioning that, as *FLea* is designed to leverage the latest feature buffer, l won't necessarily to be fixed. On the contrary, l can be dynamically altered during training based on the performance and privacy requirements.

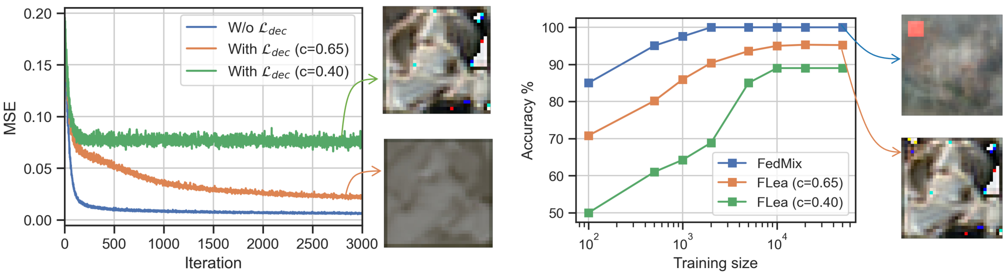
C.6 REFLECTIONS FOR BASELINES

More results for CCVR. We evaluated the baseline *CCVR* by using features from different layers to calibrate the global model, and the performance is reported in Table 9. Those results clearly suggest that leveraging features from shallower layers does not lead to further performance improvements. This suggests that post-hoc calibration has limited capability in mitigating the local drift, which is the fundamental cause of degradation in FL on non-IID data. Our *Flea* shows an evidently stronger performance.

More reflection for FedNTD. From both Figure 1 and Table 6, we can see *FedNTD* is a strong baseline for both data scarcity and label skew. *FedNTD* was devised to address the non-IID setting, but we find it is also able to alleviate issues with data scarcity in the IID setting. This suggests global knowledge distilling can mitigate local over-fitting. However, as the data becomes scarce, the distillation ability declines, herein the performance gain drops. Instead of using local data for knowledge distilling, in *FLea*, we leverage the augmented features to distil the knowledge from the global model into the local model.

Table 10: Architecture of decoder of MobileNet_V2.

Layer Index	#Input	Operator	#Output Channel	#Kernel	#Stride	#Output
1	$16 \times 32 \times 32$ (Feature)	conv2d	32	1	1	$32 \times 32 \times 32$
2	$32 \times 32 \times 32$	ConvTranspose2d	32	3	2	$32 \times 64 \times 64$
3	$32 \times 64 \times 64$	conv2d	32	3	2	$32 \times 32 \times 32$
4	$32 \times 32 \times 32$	conv2d	3	1	1	$3 \times 32 \times 32$ (Data)



(a) Reconstruction training error.

(b) Context detection accuracy.

Figure 13: The effectiveness privacy protection. c is short for the correlation in Figure 12. We show the reconstruction and context detection performance for $c = 0.65$ (the 1^{st} round) and $c = 0.40$ (the 10^{th} round).

.

D PRIVACY STUDY

Now we present the experimental setup for privacy attacks. We use the Quantity(3) data splits when $|K| = 100$ as an example for studying, as in other settings either the label is more skewed or the local data is more scarce, privacy attack can hardly be more effective than this setting. This is to present the attack defending for the most vulnerable case. As the correlation between the features and the data is continuously reduced (shown in Figure 12), we report the reconstruction and context detection performance for $c = 0.65$ (the 1^{st} round) and $c = 0.40$ (the 10^{th} round) for reference.

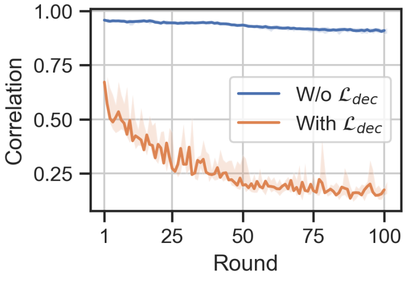


Figure 12: Correlations in each communication round.

Data reconstruction. We first implemented a data reconstruction attacker, following the approach described in Dosovitskiy & Brox (2016), the attacker constructed a decoder for the purpose of reconstruction. Specifically, the attacker targeted the converged global model trained using the Quantity(3) distribution, ensuring a fair comparison. The decoder architecture, designed to match the MobileNet_V2 architecture, comprised four *conv2d* layers (refer to Table 10) to reconstruct the original data from the provided features. For visualization purposes, the CIFAR10 images were cropped to a size of 32×32 pixels without any normalization. The decoder took the features extracted from the global model as input and generated a reconstructed image, which served as the basis for calculating the mean squared error (MSE).

To train the decoder, we utilized the entire CIFAR10 training set, conducting training for 20 epochs and employing a learning rate of 0.001. This approach allowed us to evaluate the fidelity of the reconstructed data and compare it with the original input, providing insights into the effectiveness of our proposed feature interpolation method. We use the testing set and the target global model ($c = 0.65$ and $c = 0.40$) to extract features for reconstruction. Figure 13(a) shows the training MSE while the exampled images are from the testing set. For $c = 0.65$, i.e., after the first round, in Figure 12, the sensitive attributes are removed (e.g., the color of the dog). After 10 rounds when $c < 0.4$, information is further compressed and the privacy protection is enhanced. Overall, with $\mathcal{L}_{dec}$, the correlation between data and features is reduced, preventing the image from being reconstructed.

Identifying context information. In this attack, we assume that the attacker explicitly knows the context information and thus can generate large amounts of negative (clear data) and positive (clear data with context marker) pairs to train a context classifier (which is very challenging and unrealistic but this is for the sake of testing). Real-world attack will be far more challenging than our simulations.

The context identification attacker is interested in finding out if a given feature f, is from the source data with a specific context or not. We simulate the context information by adding a color square to the image (to mimic camera broken), as illustrated in Figure 5.3. We use a binary classifier consisting of four linear layers to classify the flattened features or images. To train the classifier, we add the context marker to half of the training set. To report the identification performance, we add the same marker to half of the testing set. In Figure 13(b), the identification accuracy for *FedMix* and our *FLea* are given. We measure the attacking difficulty by how many training sample the model need to achieve a certain accuracy. The results in Figure 13(b) suggest that *FLea* needs times of training sample than *FedMix* for different correlations. This demonstrates that *FLea* can better protect the context privacy.

All the above results lead to the conclusion that by reducing feature exposure and mitigating the correlation between the features and source data, *FLea* safely protect the privacy associated with feature sharing while achieving favorable performance gain in addressing the label skew and data scarcity simultaneously.