Supplementary Materials: One-Shot Sequential Federated Learning for Non-IID Data by Enhancing Local Model Diversity

1 Experimental Setup

Adaptation Details In an effort to ensure a fair comparison, we adapted the decentralized algorithms DFedAvgM, DFedSAM, and FedSeq to the one-shot setting, and adjusted these methods to select all clients for both training and communication to fit the one-shot setting. This modification means we operate these methods for only one round of communication and select all clients for training and model distribution instead of just a portion, ensuring that all the clients' local data are utilized.

Implementation Details In all methods, for local model training, we select the model with the highest validation accuracy as the final model. We employed the Adam optimizer (excluding DFedSAM, which utilized the SAM optimizer) with a learning rate of $\eta = 5\times10^{-5}$ applicable for the CIFAR-10, PACS, and Tiny-ImageNet datasets. Due to the relatively smaller size of the Office-Caltech-10 dataset, we adopted the Adam optimizer with a learning rate of $\eta = 0.001$. For the MA-Echo method, we used the learning rate of 0.01 for all datasets as specified in their paper. In all cases, the Adam optimizer's weight decay was fixed at 1×10^{-4}. As for the other hyperparameters of the baselines, we adhered to the values suggested in their papers, which were kept constant and chosen based on optimal validation accuracy. We applied the optimal hyperparameters obtained through the grid-search strategy for our method. Specifically, we set E_{w} to 30 for the CIFAR-10 dataset, 20 for all other datasets; we set S to 5 for the CIFAR-10 dataset, 10 for the PACS and Office-Caltech-10 dataset, and 3 for the Tiny-ImageNet dataset; we set α to 0.06 for the CIFAR-10 dataset, 1 for the PACS and Tiny-ImageNet dataset, 0.001 for the Office-Caltech-10 dataset; we set β to 1 for the CIFAR-10, PACS and Tiny-ImageNet dataset, 0.001 for the Office-Caltech-10 dataset. During the local training phase of every client, in addition to utilizing scale hyperparameters α and β, our method also included control over the magnitudes of the distance parameters d_1 and d_2 to one order of magnitude smaller than the original loss ℓ, which was achieved through the logarithmic scaling. For instance, assuming the initial loss ℓ is 6.02 and the original distance d_1 is 45, we would first normalize d_1 to 0.45. Following this, we multiply this normalized distance by the scaling factor α and subsequently integrate it into the comprehensive loss function $\mathcal{L}$. This calibration ensures the original loss maintains primacy in influencing the training process, while the distance regularization terms play an ancillary role, thereby preventing their dominance over the central objective.

Environment All our experiments were conducted on a single machine with 1TB RAM and 256-core AMD EPYC 7742 64-Core Processor @ 3.4GHz CPU. The GPU we used is NVIDIA A100 SXM4 with 40GB memory. The software environment settings are: Python 3.7.4, PyTorch 1.13.1 with CUDA 11.6 on Ubuntu 20.04.4 LTS. All the experimental results are the average over three trials.

Algorithm 1: Few-Shot Adaptation of FedELMY

Input: Local datasets $\mathcal{D} = \{D_i\}_{i=1}^{N}$, warm-up epoch E_{w}, learning rate η, number of local iterations E_{local}, model number to be trained per client S, scale hyperparameters α, β

Output: The final model m_{final}

Initialization: For client 1, warm up a randomly initialized model m_{avg}^{0} for E_{w} epochs

for *shot* $r = 1 : T$ **do**
  for *client* $i = 1 : N$ **do**
    Receives m_{avg}^{i-1} from client $i-1$ (for client 1 from shot $r > 1$, receives from client N)
    // Initialize model pool $\mathcal{M}^i$ for client i
    $\mathcal{M}^i = \{m_0^i\}$ with $m_0^i \leftarrow m_{avg}^{i-1}$
    for $j = 1 : S$ **do**
      // Initialize m_j^i
      $m_j^i \leftarrow \frac{1}{|\mathcal{M}^i|} \sum_{t=0}^{|\mathcal{M}^i|-1} m_t^i$
      // Local training for m_j^i
      for $k = 1 : E_{local}$ **do**
        $\mathcal{L}(m_j^i) \leftarrow \ell(m_j^i; D_i) - \alpha \cdot d_1 + \beta \cdot d_2$
        $m_j^i \leftarrow m_j^i - \eta \nabla_m \mathcal{L}(m_j^i)$
      end
      $\mathcal{M}^i \leftarrow \mathcal{M}^i \cup \{m_j^i\}$
    end
    $m_{avg}^i \leftarrow \frac{1}{|\mathcal{M}^i|} \sum_{t=0}^{|\mathcal{M}^i|-1} m_t^i$
    Sends m_{avg}^i to client $i+1$ (for client N, sends to client 1)
  end
end
// For the final client $i = N$, outputs model m_{final}
$m_{final} \leftarrow m_{avg}^{N}$

2 Few-Shot Adaptation

Alg. 1 illustrates the implementation details and overview of our method in few-shot scenarios. In this scenario, after the local training of the final client N, it will send its averaged model m_{avg}^{N} to the first client 1 to start a new cycle of model training, this process will be repeated for multiple rounds. After completing $r = T$ iterations, we regard m_{avg}^{N} as the final global model m_{final}.

3 Adaptation to Decentralized Parallel Federated Learning

Our method can also be adapted to the decentralized Parallel Federated Learning (PFL) setting. Alg. 2 demonstrates the implementation details of our method under the decentralized PFL environment. As we can see, under this setting, clients will train their models

Algorithm 2: FedELMY for Decentralized PFL

Input: Local datasets $\mathcal{D} = \{D_i\}_{i=1}^{N}$, warm-up epoch E_{w}, learning rate η, number of local iterations E_{local}, model number to be trained per client S, scale hyperparameters α, β

Output: The final model m_{final}

Initialization: For every client i, warm up a randomly initialized model m_0^i for E_{w} epochs

for *client* $i = 1 : N$ ***in parallel*** **do**

  // Initialize model pool $\mathcal{M}^i$ for client i

  $\mathcal{M}^i = \{m_0^i\}$

  for $j = 1 : S$ **do**

    // Initialize m_j^i

    $m_j^i \leftarrow \frac{1}{|\mathcal{M}^i|} \sum_{t=0}^{|\mathcal{M}^i|-1} m_t^i$

    // Local training for m_j^i

    for $k = 1 : E_{local}$ **do**

      $\mathcal{L}(m_j^i) \leftarrow \ell(m_j^i; D_i) - \alpha \cdot d_1 + \beta \cdot d_2$

      $m_j^i \leftarrow m_j^i - \eta \nabla_m \mathcal{L}(m_j^i)$

    end

    $\mathcal{M}^i \leftarrow \mathcal{M}^i \cup \{m_j^i\}$

  end

  $m_{avg}^i \leftarrow \frac{1}{|\mathcal{M}^i|} \sum_{t=0}^{|\mathcal{M}^i|-1} m_t^i$

  Sends m_{avg}^i to all other clients $j = 1 : N$, $j \neq i$

end

// For any client $i = 1 : N$, output model m_{final}

$m_{final} \leftarrow \frac{1}{N} \sum_{i=1}^{N} m_{avg}^i$

concurrently (in parallel). During the training process, each client begins with a randomly initialized model m_0^i. Upon completion of local training, each client will simultaneously transmit the locally averaged model m_{avg}^i to all other clients. Once a client i receives all models from every neighbor, it will calculate an aggregated model by averaging its own averaged model m_{avg}^i with those received averaged models as the final model m_{final}. Please note that our framework can also be adjusted for a centralized PFL environment with a central server. In this setup, each client will send its averaged model m_{avg}^i to the central server. The overall performance will be consistent with that of the decentralized PFL adaptation since the server will finally average all models in a manner identical to the decentralized setting. Experimental results of such an adaptation are shown in Sec. 4.

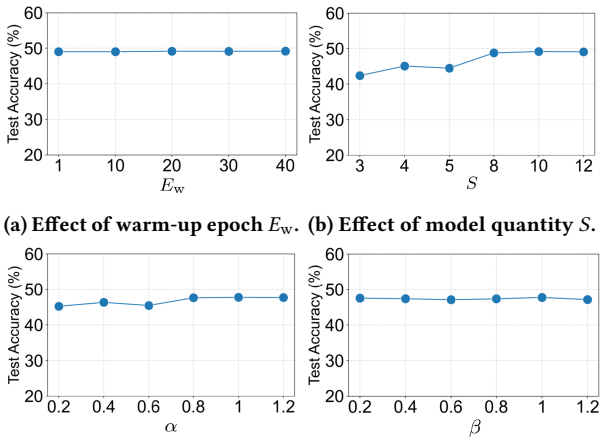
4 Additional Experiments

In the remaining parts, unless otherwise specified, we set the local training epoch E_{local} to 200, use the ResNet-18 model structure and follow the Dirichlet distribution $Dir(0.5)$ for the label-skew tasks in our experiments.

4.1 Comparison on more datasets. We compare our method to other baselines on four more public datasets, including both label-skew and domain-shift datasets to enrich our experimental results. As shown in Table 1, our method consistently outperforms all other baselines on both label-skew and domain-shift scenarios, which further validates the effectiveness of our approach.

Table 1: Test accuracy (%) comparison on more public datasets, including both label-skew and domain-shift datasets.

Distrubution	Label-Skew		Domain-Shift	
Dataset	**MNIST**	**SVHN**	**Office31**	**Office-Home**
DFedAvgM	11.35±4.01	18.60±5.66	3.47±1.11	1.71±0.23
DFedSAM	12.53±3.93	22.97±4.98	3.49±1.29	1.25±0.32
FedOV	73.10±0.33	37.47±1.48	3.39±0.46	2.31±0.15
DENSE	95.88±1.59	73.76±1.25	3.15±0.58	1.68±0.17
MetaFed	98.07±0.39	76.66±1.78	2.97±0.43	2.40±0.50
FedSeq	98.31±0.65	76.18±2.45	3.46±0.22	2.47±0.29
FedELMY	**98.91±0.23**	**79.55±4.12**	**3.57±0.09**	**2.70±0.13**



(a) Effect of warm-up epoch E_{w}. (b) Effect of model quantity S.

(c) Effect of scale parameter α. (d) Effect of scale parameter β.

Figure 1: Grid search results for PACS dataset to investigate the sensitivity of our method to different hyperparameters.

4.2 Hyperparameter sensitivity. We investigate the sensitivity of our method to different choices of hyperparameters for the PACS dataset as a supplement, as shown in Fig. 1. The findings show that our method maintains consistent robustness irrespective of the search strategies implemented, as observed across all four evaluated hyperparameters (when S grew to around 10, the performance started to become stable). Such evidence underscores the method's limited reliance on hyperparameter selection.

4.3 Impact of computation cost. We show another experiment about the effect of different computation costs on the PACS dataset with Resnet-18 structure. As we can see from Fig. 2, our method can still get the best performance under different computation cost settings, which validates the efficacy of our model training procedures by the diversity-enhanced mechanism.

4.4 Impact of client numbers. We assessed the performance of various methods across all datasets by changing the number of clients N, as shown in Table 2. It is evident that there is a general trend

Table 2: Test accuracy (%) comparison of our FedELMY method to other baselines on different numbers of clients N. For the domain-shift tasks like the PACS dataset, the number of clients $N = 8$ means we split the whole dataset into 8 parts and allocate them to the clients following the order of "*Photo* (client 1) → *Art-Painting* (client 2) → *Cartoon* (client 3) → *Sketch* (client 4) → *Photo* (client 5) → *Art-Painting* (client 6) → *Cartoon* (client 7) → *Sketch* (client 8)" for subsequent training.

Distribution	Label-Skew								Domain-Shift					
Dataset	CIFAR-10				Tiny-ImageNet				PACS			Office-Caltech-10		
N	5	20	50	100	5	20	50	100	8	20	40	8	20	40
DFedAvgM	21.27	18.22	19.50	24.48	2.64	1.96	1.99	3.09	17.55	16.37	16.62	10.65	9.73	8.94
DFedSAM	30.62	18.75	10.00	10.01	4.34	2.67	0.64	0.51	16.56	16.48	16.58	14.88	11.58	11.52
FedOV	41.49	30.93	31.93	38.42	1.33	1.23	1.11	1.08	18.59	17.73	17.32	15.78	10.56	9.34
DENSE	67.54	52.77	48.38	19.04	3.88	3.03	3.12	3.01	19.18	9.24	14.53	20.72	9.52	11.07
MetaFed	72.53	57.73	47.47	32.13	31.44	20.46	16.16	13.04	27.33	20.35	16.76	19.43	11.33	11.45
FedSeq	73.84	64.05	53.62	35.82	31.97	23.38	14.92	15.27	33.01	17.29	12.20	18.40	12.23	9.78
FedELMY	**80.99**	**68.18**	**64.31**	**40.01**	**38.43**	**29.52**	**19.93**	**20.83**	**36.10**	**25.19**	**23.60**	**24.45**	**12.48**	**12.21**

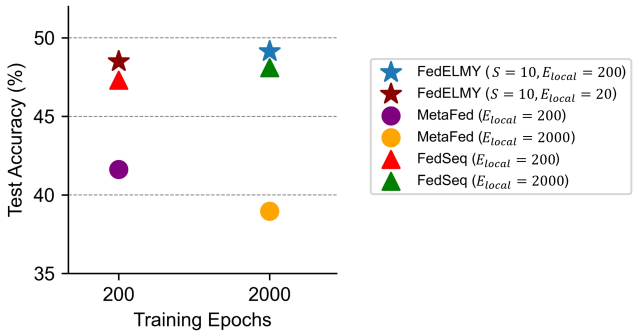


Figure 2: Test accuracy comparison with different computation costs on PACS dataset.

Table 3: Test accuracy (%) comparison of our FedELMY method to other baselines with the CNN model structure.

Dataset	CIFAR-10	Tiny-ImageNet	PACS	Office-Caltech-10
DFedAvgM	15.08	3.56	20.56	22.84
DFedSAM	13.25	0.49	15.79	9.20
FedOV	53.49	0.69	12.17	22.65
DENSE	61.42	6.72	27.16	34.88
MetaFed	56.11	6.68	28.46	29.99
FedSeq	66.08	14.49	38.36	38.22
FedELMY	**69.49**	**17.05**	**39.89**	**39.77**

toward diminishing accuracy for all methods as the client number N increases, aligning with findings reported in other literature on traditional federated learning. However, it is noteworthy that despite the influence of client quantity on one-shot federated learning, our approach consistently surpasses other baseline methods in performance. To conclude, these experiments consistently confirm the effectiveness and robustness of our proposed method to deal with one-shot sequential federated learning.

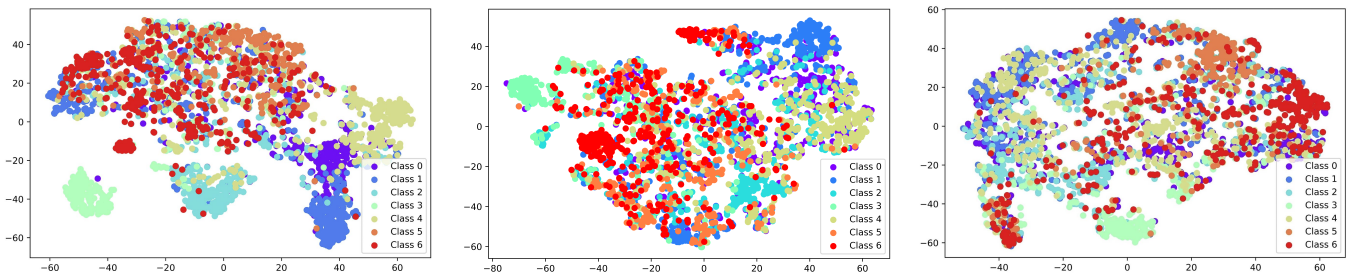
Table 4: Test accuracy (%) comparison of FedELMY to other baselines for different Dirichlet distributions.

Dataset	CIFAR-10			Tiny-ImageNet		
$Dir(\cdot)$	0.1	0.3	0.5	0.1	0.3	0.5
DFedAvgM	10.33	16.14	18.50	1.07	1.68	2.04
DFedSAM	15.62	21.84	19.48	1.74	2.80	3.41
FedOV	25.92	31.93	45.69	0.94	1.23	1.59
DENSE	26.53	61.73	64.33	1.94	2.37	1.48
MetaFed	27.18	57.24	71.29	9.96	19.45	24.53
FedSeq	27.41	57.91	73.54	15.00	22.23	24.74
FedELMY	**29.13**	**66.32**	**80.08**	**16.76**	**26.85**	**30.49**

4.5 Impact of model structure. To evaluate the scalability of our approach, we modified the model structure to a three-layer convolutional neural network (CNN) and replicated the corresponding experiments across all datasets. The results of these tests, which compare the test accuracy of our method to various baseline methods under different scenarios, are presented in Table 3, consistent with the primary analyses reported in the main paper. These findings affirm the robustness and scalability of our proposed method.

4.6 Impact of data distribution. To investigate the label-skew scenarios using the CIFAR-10 and Tiny-ImageNet datasets, we performed experiments across varying levels of skew by manipulating the Dirichlet distribution, denoted as $Dir(\cdot)$. The results, detailed in Table 4, reveal that our approach outperforms other competing methods in all scenarios, which demonstrates considerable advantages of our method, particularly in terms of scalability and privacy preservation, marking it as a more robust solution in these contexts.

4.7 PFL Adaptation. Although our method is not exclusively designed for PFL, we have nevertheless adjusted our algorithm to fit the decentralized PFL setup for a fair comparison. As shown in Table 5, it is evident that our method, even when adapted to the decentralized PFL structure, secures optimal results on most of the datasets relative to other baselines (we compare with **MA-Echo** [1], which represents the state-of-the-art one-shot decentralized PFL method; DENSE and FedOV which do not conform to the decentralized structure, are excluded from the comparison). It also surpasses



(a) FedELMY **(b) MetaFed** **(c) FedSeq**

Figure 3: T-SNE comparison between FedELMY and other SFL baselines on PACS dataset with model ResNet-18.

Table 5: Test accuracy (%, mean±std) comparison of our method under the decentralized PFL setting to other similar baselines with the Resnet-18 model structure.

Dataset	CIFAR-10	Tiny-ImageNet	PACS	Office-Caltech-10
MA-Echo	10.03±1.38	0.51±0.02	16.70±0.60	11.67±2.39
DFedAvgM	18.59±1.65	2.02±0.20	21.58±2.28	10.01±0.70
DFedSAM	18.51±1.28	3.15±0.29	20.79±1.36	**15.09±1.04**
FedELMY (PFL)	**24.71±2.62**	**4.12±0.94**	**24.13±1.57**	12.66±1.36

most existing methods on the Office-Caltech-10 dataset. Therefore, our method successfully adapts to various settings and data distributions, highlighting its broad applicability.

4.8 Classification result visualization. Fig. 3 illustrates the t-SNE plots of the feature maps for our method and other SFL baselines on the PACS dataset with model ResNet-18. As we can see, in comparison to MetaFed and FedSeq, FedELMY more effectively clusters samples of the same class, indicating a clear separation in the feature space for class 0, 1, 2, 3, and 4—a capability not observed in other baselines. Consequently, by refining the diverse local training process, our approach is adept at learning better feature representations, thereby improving overall model performance.

References

[1] Shangchao Su, Bin Li, and Xiangyang Xue. 2023. One-shot Federated Learning without server-side training. *Neural Networks* 164 (2023), 203–215. https://doi.org/10.1016/j.neunet.2023.04.035