

# No Fear of Classifier Biases: Neural Collapse Inspired Federated Learning with Synthetic and Fixed Classifier

Zexi Li<sup>1</sup>   Xinyi Shang<sup>2†</sup>   Rui He<sup>1</sup>   Tao Lin<sup>3\*</sup>   Chao Wu<sup>1\*</sup>

<sup>1</sup>Zhejiang University   <sup>2</sup>Xiamen University   <sup>3</sup>Westlake University

{zexi.li, ruihe, chao.wu}@zju.edu.cn   shangxinyi@stu.xmu.edu.cn   lintao@westlake.edu.cn

## Abstract

*Data heterogeneity is an inherent challenge that hinders the performance of federated learning (FL). Recent studies have identified the biased classifiers of local models as the key bottleneck. Previous attempts have used classifier calibration after FL training, but this approach falls short in improving the poor feature representations caused by training-time classifier biases. Resolving the classifier bias dilemma in FL requires a full understanding of the mechanisms behind the classifier. Recent advances in neural collapse have shown that the classifiers and feature prototypes under perfect training scenarios collapse into an optimal structure called simplex equiangular tight frame (ETF). Building on this neural collapse insight, we propose a solution to the FL’s classifier bias problem by utilizing a synthetic and fixed ETF classifier during training. The optimal classifier structure enables all clients to learn unified and optimal feature representations even under extremely heterogeneous data. We devise several effective modules to better adapt the ETF structure in FL, achieving both high generalization and personalization. Extensive experiments demonstrate that our method achieves state-of-the-art performances on CIFAR-10, CIFAR-100, and Tiny-ImageNet. The code is available at <https://github.com/ZexiLee/ICCV-2023-FedETF>.*

## 1. Introduction

Federated learning (FL) [33, 29, 49, 27] is a distributed training paradigm that enables collaborative training from massive multi-source datasets without transferring the raw data, reserving data ownership [30] while relieving communication burdens [33]. FL facilitates broad applications in medical images [2, 8], the internet of things [18, 34],

mobile services [9, 17], and so on; it shows promising prospects in data collaboration. However, clients in FL training may hold heterogeneous data, in other words, clients’ datasets are in Non-IID distributions<sup>1</sup>, which causes a huge degradation to the global model’s generalization [5, 32, 3, 28, 14, 15].

Numerous recent studies have shown that **classifier biases** in clients’ local models caused by Non-IID data are the primary cause of degradation in FL [32, 53, 22]. It has been discovered that the classifier layer is more biased than other layers [32], and classifier biases will create a *vicious cycle* between biased classifiers and misaligned features across clients [53]. Figure 1 illustrates the issue of classifier bias in FL, where Non-IID data leads to poor pairwise cosine similarities among clients’ classifiers and feature prototypes. Furthermore, class-wise classifier vectors of clients are scattered in the embedding space, leading to significant generalization declines.

Previous research has attempted to mitigate classifier biases through classifier retraining via generated virtual features at the end of FL training [32, 38]. However, these methods fail to address classifier biases during training. Biased classifiers during the training phase lead to inadequate feature extractors and poor representations of generated features, negatively affecting the retrained classifiers. Our experiments have also shown the limitations of classifier retraining methods (Table 1). Therefore, we wonder:

*Can we break the classifier bias dilemma during training, improving both the classifiers and feature representations?*

To resolve the classifier bias dilemma, it is essential to fully understand the mechanisms behind the classifier. We further wonder: *what are the properties of a well-trained (i.e. good) classifier?* An emerging discovery called neural collapse [35, 44, 26, 45] has shed light on this matter. It describes the phenomenon that, in the perfect training sce-

\*Corresponding authors. <sup>†</sup>Work was done during Xinyi’s visit to Westlake University.

<sup>1</sup>We use “data heterogeneity” and “Non-IID data” interchangeably.

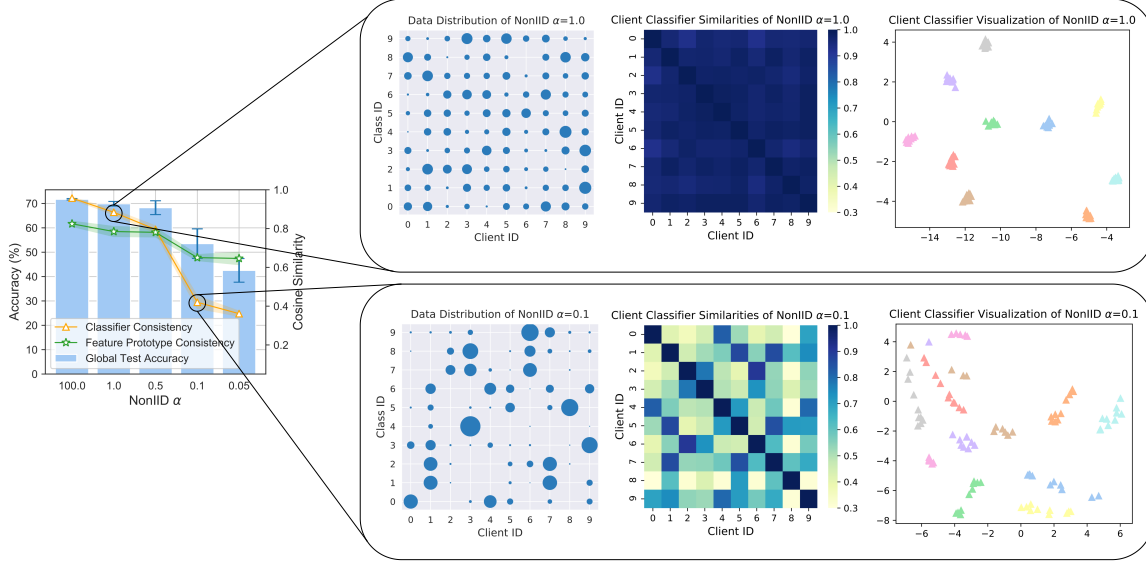


Figure 1. **How data heterogeneity causes classifier biases in FL.** Smaller  $\alpha$  corresponds to higher Non-IID. Experiments are conducted on CIFAR-10 with vanilla FEDAVG. Columns from left to right: (1) Non-IID data results in poor generalization, biased classifiers, and misaligned features. (2) Clients’ data distributions. (3) Clients with Non-IID data have smaller pair-wise classifier cosine similarities. (4) t-SNE visualization of clients’ class-wise classifier vectors (represented by colors), which are more scattered in Non-IID data.

nario, where the dataset is balanced and sufficient, the feature prototypes and classifier vectors converge to an optimal simplex equiangular tight frame (ETF) with maximal pairwise angles [35]. Insights from balanced training inspire us to tackle the challenges in Non-IID FL.

Thus, in this paper, *we fundamentally solve the FL’s classifier bias problem with a neural-collapse-inspired approach.* Knowing the optimal classifier structure, we make the first attempt to introduce a synthetic simplex ETF as a fixed classifier for all clients so that the clients can learn unified and optimal feature representations even under high heterogeneity. We devise FEDETF which incorporates several effective modules that better adapt the ETF structure in FL training, reaching strong results on *both generalization and personalization*.

Specifically, we employ a projection layer that maps the features to a space where neural collapse is more likely to occur. We also implement a balanced feature loss with a learnable temperature to minimize entropy between the features and the fixed ETF classifier. These techniques enable us to achieve high generalization performance of the global model during FL training. To further improve personalization, we introduce a novel fine-tuning strategy that adapts the global model locally after FL training. Extensive experiments have strongly supported the effectiveness of our method. Our contributions are summarized as follows.

- To the best of our knowledge, this is the first paper that tackles the data heterogeneity problem in FL from the perspective of neural collapse.

- We devise FEDETF, which takes the simplex ETF as a fixed classifier, and it fundamentally solves the classifier biases brought by Non-IID data, reaching high generalization of the global model.
- We propose a local fine-tuning strategy in FEDETF to boost personalization in each client after FL training.
- Our method is validated on three vision datasets: CIFAR-10, CIFAR-100, and Tiny-ImageNet. Our proposed method outperforms strong baselines and achieves sota in both generalization and personalization.

## 2. Related Works

**Data Heterogeneity in Federated Learning.** A variety of solutions have been proposed to tackle data heterogeneity in FL. Recent works [22, 32, 38, 53] have revealed that the biased classifier is the main cause leading to poor performance of the global model, and they use classifier retraining [32, 38] or classifier variance reduction [22] to calibrate the classifier. In particular, CCVR [32] finds that there exists a greater bias in the classifier than in other layers, and only calibrating the classifier via virtual features after FL training can improve the global model performance. However, this approach cannot resolve the misaligned representations of local models caused by biased classifiers during FL training. Consequently, the backbone feature extractor cannot be improved. Moreover, some concurrent works use classifier variance reduction [22] or feature anchors [53] to relieve classifier biases. However, when data is highly Non-IID, variance-reduced classifiers and aggregated feature anchors

are also biased and far from optimal (i.e., trained on IID data). Although these methods can alleviate classifier biases to some extent, they cannot completely solve them.

To improve the misaligned features, another line of works use prototypical methods to aid client training. FED-PROTO [40] only transmits and aggregates prototypes on the server to deal with data heterogeneity and model heterogeneity, and FEDFM anchors clients' features to improve generalization [46]. A concurrent work named FEDNH [5] adopts a prototypical classifier and uses a smoothing aggregation strategy to update the classifier based on clients' local prototypes. However, the aggregated prototypes will also be biased in extreme Non-IID settings, and these methods did not use the classifier's ETF optimality to tackle this problem, which is where our contribution lies.

Besides, data heterogeneity can be relieved by improving aggregation [28, 47]. Apart from generalization, there are data heterogeneity challenges for personalization, methods such as decoupling classifiers and feature extractors [3], separating feature information [51], adaptive local aggregation [50], and edge-cloud collaboration [27] can be used to facilitate better local personalized models.

**Neural Collapse.** The neural collapse was firstly observed in [35] that at the terminal phase of training on a balanced dataset, the feature prototypes and the classifier vectors will converge to a simplex ETF where the vectors are normalized and the pair-wise angles are maximized. Afterward, there are some works trying to figure out the mechanism behind neural collapse [16, 54, 42, 19] and in which conditions neural collapse will happen [26, 42, 19]. Recent works use neural-collapse-inspired methods to solve the problems in imbalanced training [44, 43, 41], incremental learning [45], and transfer learning [26]. Despite the neural-collapse-inspired methods' success in centralized learning, deep insights and effective solutions regarding neural collapse are missing in distributed training. In this paper, we find neural collapse is also the key to success in FL and we show that inducing ETF optimality can inherently solve the classifier bias and feature misalignment problem in FL and largely improve performance.

### 3. Preliminaries

#### 3.1. Federated Learning

**Basic Settings.** We introduce a typical FL setting with  $K$  clients holding potentially Non-IID data partitions  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$ , respectively. A supervised classification task with  $C$  classes is considered. Let  $n_{k,c}$  be the number of samples of class  $c$  on client  $k$ , and  $n_k = \sum_{c=1}^C n_{k,c} = |\mathcal{D}_k|$  denotes the number of training samples held by client  $k$ . FL aims to realize generalization or personalization by distributed training under the coordination of a central server without any data transmission. For *generalization* in FL, the

goal is to learn a global model over the whole training data  $\mathcal{D} \triangleq \bigcup_k \mathcal{D}_k$  and the global model is expected to be generalized to the distribution of the whole data  $\mathcal{D}$ . For *personalization*, the goal is to learn  $K$  personalized local models by FL training, and the local model  $k$  is expected to have better adaptation in local data distribution  $\mathcal{D}_k$  than the independently trained one. For the model in FL, we typically consider a neural network  $\phi_{\mathbf{w}}$  with parameters  $\mathbf{w} = \{\mathbf{u}, \mathbf{v}\}$ . It has two main components: 1) a feature extractor  $f_{\mathbf{u}}$  parameterized by  $\mathbf{u}$ , mapping each input sample  $\mathbf{x}$  to a  $d$ -dim feature vector; 2) a classifier  $h_{\mathbf{v}}$  parameterized by  $\mathbf{v}$ . The parameters of client  $k$ 's local model are denoted as  $\mathbf{w}_k$ .

**Model updates.** There are two iterative steps in FL, client local training and server global aggregation, and the iteration lasts for  $T$  rounds. In round  $t$ , the server first sends a global model  $\mathbf{w}^t$  to clients.

**Client local training:** For each client  $k$ ,  $\forall k \in [K]$ , it conducts SGD updates on the local data  $\mathcal{D}_k$ :

$$\mathbf{w}_k^t \leftarrow \mathbf{w}_k^t - \eta \nabla_{\mathbf{w}} \ell(\mathbf{w}_k^t; \mathcal{B}^i), \quad (1)$$

where  $\eta$  is the learning rate,  $\mathcal{B}^i$  is the mini-batch sampled from  $\mathcal{D}_k$  at the  $i$ -th local iteration. The local epoch is  $E$ .

**Server global aggregation:** After local training, the server samples a set of clients  $\mathcal{A}^t$  and the sampled clients send their updated models to the server. Then the server performs the weighted aggregation to update the global model for round  $t + 1$ . The vanilla aggregation strategy is FEDAVG [33] where the aggregation weights are proportional to clients' data sizes.

$$\mathbf{w}^{t+1} = \sum_{k \in \mathcal{A}^t} \frac{n_k}{\sum_{j \in \mathcal{A}^t} n_j} \mathbf{w}_k^t. \quad (2)$$

#### 3.2. Neural Collapse

Neural collapse refers to a phenomenon about the last-layer features and classifier vectors at the terminal phase of training (zero training loss) on a balanced dataset [35]. We first give the definition of simplex ETF in neural collapse.

**Definition 3.1** (Simplex Equiangular Tight Frame). A collection of vectors  $\mathbf{v}_i \in \mathbb{R}^d$ ,  $i \in [C]$ ,  $d \geq C - 1$ , is said to be a simplex equiangular tight frame if:

$$\mathbf{V} = \sqrt{\frac{C}{C-1}} \mathbf{U} \left( \mathbf{I}_C - \frac{1}{C} \mathbf{1}_C \mathbf{1}_C^T \right), \quad (3)$$

where  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_C] \in \mathbb{R}^{d \times C}$ ,  $\mathbf{U} \in \mathbb{R}^{d \times C}$  allows a rotation and satisfies  $\mathbf{U}^T \mathbf{U} = \mathbf{I}_C$ ,  $\mathbf{I}_C$  is the identity matrix, and  $\mathbf{1}_C$  is an all-ones vector. All vectors in a simplex ETF have an equal  $\ell_2$  norm and the same pair-wise angle, i.e.

$$\mathbf{v}_i^T \mathbf{v}_j = \frac{C}{C-1} \delta_{i,j} - \frac{1}{C-1}, \forall i, j \in [C], \quad (4)$$

where  $\delta_{i,j}$  equals to 1 when  $i = j$  and 0 otherwise. The pair-wise angle  $-\frac{1}{C-1}$  is the maximal equiangular separation of  $C$  vectors in  $\mathbb{R}^d$  [35].

We highlight three key properties of the neural collapse (NC) phenomenon below.

**NC1** (Features collapse to the class prototypes). The last-layer features will collapse to their within-class mean (prototypes), i.e. for any class  $c$ ,  $\forall c \in [C]$ , the covariance  $\Sigma_W^c \rightarrow \mathbf{0}$ , where  $\Sigma_W^c := \frac{1}{n_c} \sum_{i=1}^{n_c} (\mathbf{h}_{c,i} - \mathbf{h}_c)(\mathbf{h}_{c,i} - \mathbf{h}_c)^T$ .  $\mathbf{h}_{c,i} = f(\mathbf{u}; \mathbf{x}_{c,i})$  is the feature of the  $i$ -th sample in the class  $c$ , and  $\mathbf{h}_c = \frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{h}_{c,i}$  is the class  $c$ 's prototype.

**NC2** (Prototypes collapse to simplex ETFs).  $\tilde{\mathbf{h}}_c = (\mathbf{h}_c - \mathbf{h}_G)/\|\mathbf{h}_c - \mathbf{h}_G\|, \forall c \in [C]$ , collapses to a simplex ETF which satisfies Eq. (4).  $\mathbf{h}_G$  is the global mean of the last-layer features, that  $\mathbf{h}_G = \sum_{c=1}^C \sum_{i=1}^{n_c} \mathbf{h}_{c,i}$ .

**NC3** (Classifiers collapse to the same simplex ETFs). The normalized feature prototype  $\tilde{\mathbf{h}}_c$  is aligned with their corresponding classifier weights<sup>2</sup>, which means that the classifier weights collapse to the same simplex ETF, i.e.  $\tilde{\mathbf{h}}_c = \mathbf{v}_c/\|\mathbf{v}_c\|$ , where  $\mathbf{v}_c$  refers to the vectorized classifier weights of class  $c$ .

## 4. Methods

Neural collapse tells us the optimal structure (i.e. simplex ETF) of classifiers and feature prototypes in a perfect training setting. It inspires us to use a synthetic simplex ETF as a fixed classifier from the start to mitigate the classifier bias and feature misalignment problems (see Figure 1) brought by clients' data heterogeneity. Therefore, we propose FEETF, a novel FL algorithm inspired by neural collapse. Concretely, as elaborated in Section 4.1, to promote the generalization of the global model, we reformulate the model architecture by replacing the learnable classifier with a fixed ETF classifier and devise a tailored loss for robust learning during FL training. Moreover, as described in Section 4.2, to improve local personalization after FL training, we propose a finetuning strategy for both finetuning the model and the formerly fixed ETF classifier.

### 4.1. Improving Generalization by ETF Classifier

**Reformulation of the model architecture.** In previous works of FL, a model architecture that consists of a learnable feature extractor and a learnable linear classifier is adopted [33, 3, 4, 25], as shown in Figure 2 (a). However, due to clients' data heterogeneity, the classifiers will be more biased than other layers [32, 22], and a vicious cycle between classifier biases and feature misalignment will exist [53]. In this paper, we reformulate the model in FL into the combination of a learnable feature extractor and a fixed ETF classifier, as demonstrated in Figure 2 (b).

**ETF classifier initialization:** At the beginning of the FL training, we first randomly synthesize a simplex ETF  $\mathbf{V}_{ETF} \in \mathbb{R}^{d \times C}$  by Eq. (3), where  $d$  denotes the feature dimension of the ETF and  $C$  is the number of classes. The

feature dimension  $d$  should require  $d \geq C - 1$  in Definition 3.1, and we will discuss in Figure 6 (a) that a relatively low dimension is beneficial to neural collapse. For each class's classifier vector  $\mathbf{v}_i, \forall i \in [C]$  in the ETF  $\mathbf{V}_{ETF}$ , it requires  $\|\mathbf{v}_i\|_2 = 1$ ; and any pair of classifier vectors  $(\mathbf{v}_i, \mathbf{v}_j), i \neq j, \forall i, j \in [C]$  satisfies  $\cos(\mathbf{v}_i, \mathbf{v}_j) = -\frac{1}{C-1}$  according to Eq. (4).

**Projection layer:** Given a data sample  $\mathbf{x}$ , we first use the feature extractor  $f_{\mathbf{u}}$  to transform the data into the raw feature  $\mathbf{h}$  and then use a projection layer  $g_{\mathbf{p}}$  to map this raw feature to the ETF feature space and normalize it into  $\boldsymbol{\mu}$ .

$$\boldsymbol{\mu} = \hat{\boldsymbol{\mu}}/\|\hat{\boldsymbol{\mu}}\|_2, \quad \hat{\boldsymbol{\mu}} = g(\mathbf{p}; \mathbf{h}), \quad \mathbf{h} = f(\mathbf{u}; \mathbf{x}), \quad (5)$$

where  $\mathbf{u}$  and  $\mathbf{p}$  denote the parameters of the feature extractor and the projection layer. We note that the projection layer is essential in our FEETF design: 1) If the last layer of the feature extractor is the non-linear activation, e.g. the ReLU, the raw feature  $\mathbf{h}$  will be sparse with zeros (or near zero values), and it is hard for  $\mathbf{h}$  to be close to the dense ETF classifier vectors. 2) The raw features always have high dimensions, and high-dimensional vectors are more prone to be orthogonal, which is harder to collapse into the ETF with maximal angles. It is necessary to use the projection layer to map the features into a suitable dimension  $d$ . 3) The projection layer is helpful in the local finetuning stage for personalization.

**Balanced feature loss with learnable temperature.** In neural-collapse-inspired imbalanced learning [48], it is found that when the ETF classifier is used, the gradients of cross entropy (CE) will be biased towards the head class, and the authors proposed a dot regression loss to tackle this problem. In FL, clients' local datasets are also class-imbalanced due to data heterogeneity, so techniques tackling the imbalanced problem are also needed in our design. Following previous work [3] which induces balanced softmax loss to logit-prediction-based CE in FL, in this paper, we also incorporate the balanced loss [36] into our feature-based CE. Instead of using former dot regression loss [48, 45], it is found that our balanced feature CE loss also can solve the imbalanced gradient problem in learning with the ETF classifier.

Moreover, the softmax function's input of the vanilla CE loss is the logits, generated by MLP, while that of our method is the features' product  $\mathbf{v}_y^T \boldsymbol{\mu}$ . The logits have a wide range of value since the output of MLP has no constraints, but our features' product has a limited range  $[-1, 1]^3$ , which is sensitive to scaling. Therefore, we add a temperature<sup>4</sup> scalar  $\beta$  to scale the features' product. Further, we found that in different stages of training, it requires different  $\beta$ , and fixed  $\beta$  is hard to tune and may impede performance

<sup>3</sup>Knowing the fact that both the vectors are normalized.

<sup>4</sup>The term "temperature" is borrowed from a similar concept in knowledge distillation [11].

<sup>2</sup>For simplicity, we omit the bias term in a linear classifier layer.

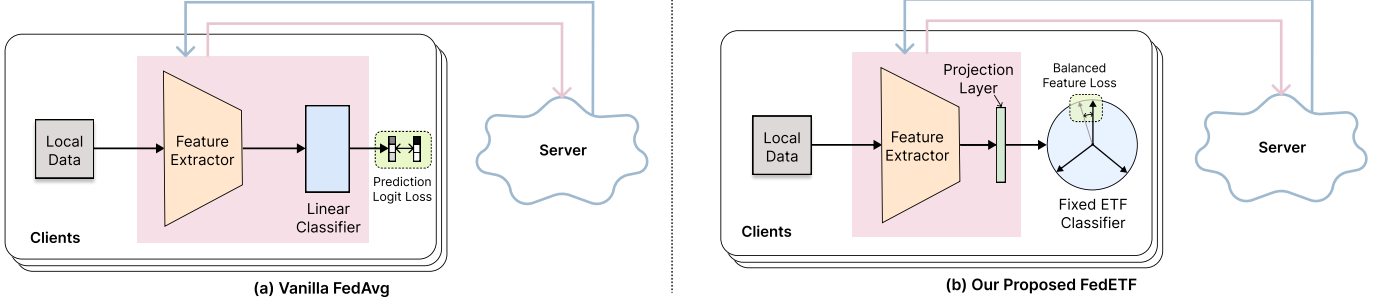


Figure 2. **Proposed FEDETF during FL training.** (a) In vanilla FL training, the feature extractor and linear classifier are both learned at clients and aggregated at server. (b) In our FEDETF, only the feature extractor and projection layer are learned and aggregated, and we adopt the same synthetic and fixed ETF classifier for all clients throughout the FL training process. Instead of prediction logit loss in vanilla FL, we use a novel balanced feature loss for the ETF classifier.

if not appropriate. To solve this, we take  $\beta$  as one of the parameters in the model and update it by SGD during training. This learnable temperature  $\beta$  will capture the learning dynamics in each client under various heterogeneity.

We define the model parameters in our FEDETF as  $\mathbf{w} = \{\mathbf{u}, \mathbf{p}, \beta\}$ , which consists of the feature extractor, the projection layer, and the learnable temperature. For a given sample  $(\mathbf{x}, y)$  of client  $k$ ,  $\forall k \in [K]$ , we define the loss function for generalization in Eq. (6), where the **orange** term is for balanced feature loss and the **blue** term is for learnable temperature.

$$\ell^g(\mathbf{w}, \mathbf{V}_{ETF}; \mathbf{x}, y) = -\log \frac{n_{k,y}^\gamma \exp(\beta \cdot \mathbf{v}_y^T \boldsymbol{\mu})}{\sum_{c \in [C]} n_{k,c}^\gamma \exp(\beta \cdot \mathbf{v}_c^T \boldsymbol{\mu})}, \quad (6)$$

where  $n_{k,c}$  refers to the number of samples in class  $c$  of client  $k$ ,  $\beta$  is the learnable temperature,  $\boldsymbol{\mu}$  is the normalized feature in Eq. (5),  $\mathbf{v}_c$  is the class  $c$ 's classifier vector in  $\mathbf{V}_{ETF}$ , and  $\gamma$  is the hyperparameter for balanced loss. Below, we give the learning objective of client  $k$ ,  $\forall k \in [K]$ , and solve the objective by SGD in Eq. (1).

$$\mathbf{w}_k^t = \arg \min_{\mathbf{w}} \mathcal{L}_k^g(\mathbf{w}), \quad (7)$$

$$\text{where } \mathcal{L}_k^g(\mathbf{w}) = \frac{1}{n_k} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_k} \ell^g(\mathbf{w}, \mathbf{V}_{ETF}; \mathbf{x}_i, y_i). \quad (8)$$

We adopt the vanilla aggregation strategy on the server, formulated in Eq. (2). The pseudo code of the proposed FEDETF is shown in Algorithm 1 in Appendix.

## 4.2. Personalized Adaptation by Local Finetuning

As discovered in [3], local adaptation of a more generalized global model will reach stronger personalization. We will also verify this finding in our FEDETF. After the FL training, we obtain a global model  $\mathbf{w}^g$  with better generalization, and we use  $\mathbf{w}^g$  as the initialization in each client for personalization. We will show that by our tailored local finetuning of  $\mathbf{w}^g$ , we will also reach the state-of-the-art in personalization.

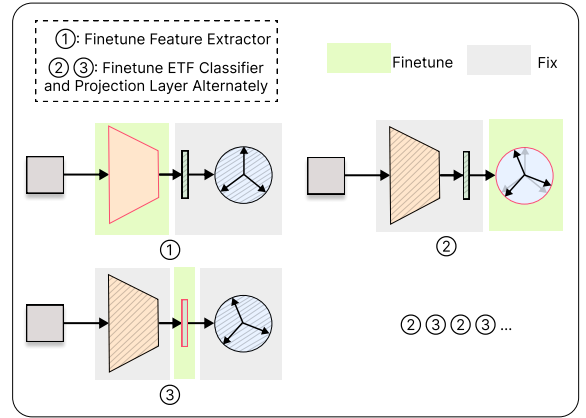


Figure 3. **Local finetuning stage of proposed FEDETF for personalization.** This stage is after the whole FL training stage when clients receive the final global model. For each client, we first finetune the feature extractor and then we finetune the ETF prototypical classifier and projection layer alternately.

Our personalized local finetuning consists of two parts: *local feature adaptation* and *classifier finetuning*. In the *local feature adaptation*, we fix the projection layer and ETF classifier and finetune the feature extractor to let the feature extractor be more customized to the features of clients' local data. In the *classifier finetuning period*, we finetune the ETF classifier and projection layer alternately for several iterations to make the classifier more biased to the local class distributions. We note that the simplex ETF is not an ideal classifier for local personalization, since the clients may have imbalanced class distributions or even have missing classes. Biased classifiers are needed for personalization to take the local class distributions as prior knowledge and maximize the prediction likelihood. We alternately finetune the ETF classifier and projection layer to make the projected features and classifier vectors converge to be aligned.

The process of local finetuning is illustrated in Figure 3. The learned model parameters in the personalization stage are  $\mathbf{w} = \{\mathbf{u}, \mathbf{p}, \beta, \mathbf{V}_{ETF}\}$ , and we split  $\mathbf{w}$  into the tuned

Table 1. **Results in terms of generalization (General.) accuracy (%) of global models and personalization (Personal.) accuracy (%) of local models on three datasets under different heterogeneity.** Best two methods in each setting are highlighted in **bold** fonts.

Dataset	CIFAR-10				CIFAR-100				Tiny-ImageNet			
NonIID ( $\alpha$ )	0.1		0.05		0.1		0.05		0.1		0.05	
Methods/Metrics	General.	Personal.	General.	Personal.	General.	Personal.	General.	Personal.	General.	Personal.	General.	Personal.
FEDAVG [33]	52.76 $\pm$ 6.08	83.85 $\pm$ 0.89	44.48 $\pm$ 6.19	89.80 $\pm$ 0.39	24.77 $\pm$ 1.19	49.93 $\pm$ 1.17	<b>22.53<math>\pm</math>0.40</b>	58.85 $\pm$ 0.33	28.93 $\pm$ 0.52	40.81 $\pm$ 0.35	24.88 $\pm$ 0.34	46.90 $\pm$ 0.44
FEDPROX [25]	46.59 $\pm$ 3.04	82.08 $\pm$ 0.27	40.95 $\pm$ 5.75	87.69 $\pm$ 2.85	23.33 $\pm$ 1.72	46.44 $\pm$ 1.64	19.12 $\pm$ 0.77	57.01 $\pm$ 2.17	25.93 $\pm$ 0.27	31.90 $\pm$ 1.91	23.06 $\pm$ 0.68	32.43 $\pm$ 0.65
FEDDYN [1]	36.35 $\pm$ 5.33	85.39 $\pm$ 0.77	23.90 $\pm$ 1.40	88.72 $\pm$ 1.59	<b>25.53<math>\pm</math>2.39</b>	51.79 $\pm$ 2.12	20.71 $\pm$ 2.83	<b>61.77<math>\pm</math>0.32</b>	26.42 $\pm$ 0.56	45.84 $\pm$ 0.34	23.63 $\pm$ 1.55	52.27 $\pm$ 1.06
DITTO [24]	52.76 $\pm$ 6.08	79.81 $\pm$ 1.89	44.48 $\pm$ 6.19	85.17 $\pm$ 3.47	24.77 $\pm$ 1.19	38.06 $\pm$ 1.26	22.53 $\pm$ 0.40	50.18 $\pm$ 1.22	28.93 $\pm$ 0.52	33.00 $\pm$ 1.01	24.88 $\pm$ 0.34	40.31 $\pm$ 0.12
FEDREP [4]	26.85 $\pm$ 10.13	<b>87.76<math>\pm</math>0.87</b>	15.79 $\pm$ 3.68	90.71 $\pm$ 2.25	5.47 $\pm$ 0.20	<b>53.62<math>\pm</math>1.49</b>	4.18 $\pm$ 0.85	61.51 $\pm$ 0.61	4.10 $\pm$ 0.22	43.66 $\pm$ 0.48	2.20 $\pm$ 0.19	49.52 $\pm$ 1.64
CCVR [32]	52.50 $\pm$ 6.31	55.62 $\pm$ 5.89	47.98 $\pm$ 6.24	73.52 $\pm$ 7.49	24.54 $\pm$ 0.71	34.01 $\pm$ 2.01	22.28 $\pm$ 0.43	39.16 $\pm$ 1.41	<b>32.78<math>\pm</math>0.24</b>	<b>54.00<math>\pm</math>0.46</b>	<b>29.27<math>\pm</math>0.25</b>	<b>59.29<math>\pm</math>0.30</b>
FEDPROTO [40]	-	83.34 $\pm$ 0.71	-	88.21 $\pm$ 1.77	-	43.31 $\pm$ 0.70	-	54.87 $\pm$ 0.52	-	40.74 $\pm$ 0.87	-	48.05 $\pm$ 0.82
FEDROD [3]	<b>55.72<math>\pm</math>2.40</b>	86.19 $\pm$ 0.91	<b>49.89<math>\pm</math>3.64</b>	88.83 $\pm$ 4.14	24.49 $\pm$ 1.05	51.78 $\pm$ 1.16	21.63 $\pm$ 0.42	59.44 $\pm$ 0.45	32.17 $\pm$ 0.41	38.27 $\pm$ 1.00	28.45 $\pm$ 0.58	44.09 $\pm$ 0.44
FEDNH [5]	55.37 $\pm$ 4.48	85.98 $\pm$ 0.15	47.96 $\pm$ 2.59	<b>91.06<math>\pm</math>3.13</b>	24.67 $\pm$ 0.68	52.09 $\pm$ 0.78	21.95 $\pm$ 0.85	<b>62.71<math>\pm</math>0.22</b>	17.51 $\pm$ 0.62	36.53 $\pm$ 0.29	14.00 $\pm$ 0.17	41.80 $\pm$ 1.78
<b>Our FEDETf</b>	<b>59.56<math>\pm</math>1.84</b>	<b>87.89<math>\pm</math>1.19</b>	<b>56.08<math>\pm</math>3.44</b>	<b>92.62<math>\pm</math>0.54</b>	<b>26.24<math>\pm</math>1.78</b>	<b>52.86<math>\pm</math>1.53</b>	<b>24.17<math>\pm</math>0.54</b>	60.68 $\pm$ 0.91	<b>33.49<math>\pm</math>0.82</b>	<b>55.82<math>\pm</math>0.60</b>	<b>29.15<math>\pm</math>1.03</b>	<b>62.36<math>\pm</math>0.13</b>

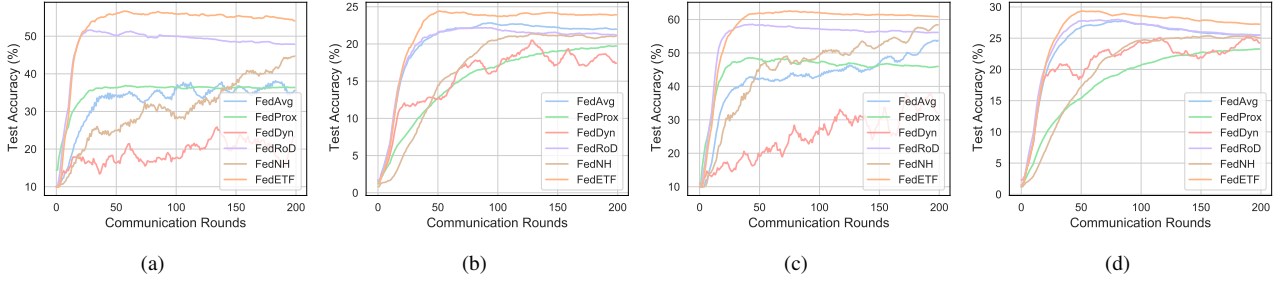


Figure 4. **Global models' test accuracy curves of the methods.** (a) CIFAR-10 with  $\alpha = 0.05$ . (b) CIFAR-100 with  $\alpha = 0.05$ . (c) CIFAR-10 with  $\alpha = 0.1$ . (d) CIFAR-100 with  $\alpha = 0.1$ .

parameters  $\hat{\mathbf{w}}$  and the fixed parameters  $\bar{\mathbf{w}}$ . When finetune the feature extractor,  $\hat{\mathbf{w}} = \{\mathbf{u}, \beta\}$ ,  $\bar{\mathbf{w}} = \{\mathbf{p}, \mathbf{V}_{ETF}\}$ ; when finetune the ETF classifier,  $\hat{\mathbf{w}} = \{\mathbf{V}_{ETF}, \beta\}$ ,  $\bar{\mathbf{w}} = \{\mathbf{u}, \mathbf{p}\}$ ; when finetune the projection layer,  $\hat{\mathbf{w}} = \{\mathbf{p}, \beta\}$ ,  $\bar{\mathbf{w}} = \{\mathbf{u}, \mathbf{V}_{ETF}\}$ . We use the vanilla CE loss without balanced softmax in each stage of finetuning.

$$\ell^p(\hat{\mathbf{w}}, \bar{\mathbf{w}}; \mathbf{x}, y) = -\log \frac{\exp(\beta \cdot \mathbf{v}_y^T \boldsymbol{\mu})}{\sum_{c \in [C]} \exp(\beta \cdot \mathbf{v}_c^T \boldsymbol{\mu})}. \quad (9)$$

The learning objective in each stage is shown as follows.

$$\mathbf{w}_k^p = \{\bar{\mathbf{w}}, \arg \min_{\hat{\mathbf{w}}} \mathcal{L}_k^p(\hat{\mathbf{w}})\}, \quad (10)$$

$$\text{where } \mathcal{L}_k^p(\hat{\mathbf{w}}) = \frac{1}{n_k} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_k} \ell^p(\hat{\mathbf{w}}, \bar{\mathbf{w}}; \mathbf{x}_i, y_i). \quad (11)$$

Personalization will be reached after several iterative stages of finetuning in Figure 3. The pseudo code of the personalized local finetuning is shown in Algorithm 2 in Appendix.

## 5. Experiments and Results

### 5.1. Settings

**Datasets and Models.** Following previous works [5, 31], we use three vision datasets to conduct experiments:

CIFAR-10 [20], CIFAR-100 [20], and Tiny-ImageNet [6, 21]. Tiny-ImageNet is a subset of ImageNet with 100k samples of 200 classes. Following [23], we adopt ResNet20 [23, 10] for CIFAR-10/100 and use ResNet-18 [23, 10] for Tiny-ImageNet. We use a linear layer as the classifier for the baselines and as the projection layer for our method.

**Compared Methods.** We take three lines of methods as baselines. **1) Classical FL with Non-IID data:** FEDAVG [33] with vanilla local training, a simple but strong baseline; FEDPROX [25], FL with proximal regularization at clients; FEDDYN [1], FL based on dynamic regularization. **2) Personalized FL:** DITTO [24], personalization through separated local models; FEDREP [4], personalization by only aggregating feature extractors; FEDROD [3], personalization through decoupling models. **3) FL methods most relevant to ours:** CCVR [32], FL with classifier retraining; FEDPROTO [40], FL with only prototype sharing; FEDROD [3], generalization through decoupling and balanced softmax loss; FEDNH [5], FL with smoothing aggregation of prototypical classifiers.

**Client Settings.** We adopt the Dirichlet sampling to generate Non-IID data for each client. We note that the Dirichlet-sampling-based data heterogeneity is widely used in FL literature [31, 3, 5, 32]. It considers a class-imbalanced data heterogeneity, controlled by hyperparameter  $\alpha$ , and smaller  $\alpha$  refers to more Non-IID data of clients. In our experi-



Table 2. **Results (%) under various numbers of clients with partial client sampling.** The dataset is CIFAR-10 with Non-IID  $\alpha = 0.1$ .

Number of Clients	50				100			
Sampling Rate	0.4		0.6		0.4		0.6	
Methods/Metrics	General.	Personal.	General.	Personal.	General.	Personal.	General.	Personal.
FEDAVG [33]	38.13 $\pm$ 5.12	77.28 $\pm$ 2.17	42.68 $\pm$ 6.28	74.99 $\pm$ 2.34	42.15 $\pm$ 1.61	71.52 $\pm$ 1.88	41.42 $\pm$ 3.31	70.40 $\pm$ 2.13
CCVR [32]	44.59 $\pm$ 11.4	<b>78.93<math>\pm</math>3.26</b>	52.49 $\pm$ 6.73	<b>82.33<math>\pm</math>1.72</b>	50.07 $\pm$ 0.80	<b>76.27<math>\pm</math>2.08</b>	50.41 $\pm$ 3.93	<b>77.27<math>\pm</math>1.22</b>
FEDROD [3]	<b>55.84<math>\pm</math>3.96</b>	76.60 $\pm$ 0.13	<b>53.04<math>\pm</math>2.54</b>	74.42 $\pm$ 1.99	<b>52.62<math>\pm</math>1.68</b>	71.27 $\pm$ 0.69	<b>52.34<math>\pm</math>0.11</b>	72.41 $\pm$ 0.74
FEDNH [5]	39.97 $\pm$ 6.90	76.59 $\pm$ 0.59	45.36 $\pm$ 3.58	78.17 $\pm$ 1.15	42.77 $\pm$ 0.65	73.47 $\pm$ 1.38	45.85 $\pm$ 2.98	73.15 $\pm$ 0.95
<b>Our FEDETf</b>	<b>58.05<math>\pm</math>4.63</b>	<b>85.82<math>\pm</math>0.86</b>	<b>58.75<math>\pm</math>1.72</b>	<b>85.05<math>\pm</math>0.87</b>	<b>56.67<math>\pm</math>0.88</b>	<b>83.47<math>\pm</math>0.45</b>	<b>55.96<math>\pm</math>0.23</b>	<b>83.38<math>\pm</math>0.72</b>

Table 3. **Results (%) under different local epochs ( $E$ ).** The dataset is CIFAR-10 with Non-IID  $\alpha = 0.1$ .

$E$	1		2		4		8	
Methods/Metrics	General.	Personal.	General.	Personal.	General.	Personal.	General.	Personal.
FEDAVG [33]	45.75 $\pm$ 1.97	74.18 $\pm$ 2.22	43.02 $\pm$ 2.16	77.45 $\pm$ 0.79	36.30 $\pm$ 2.88	80.66 $\pm$ 1.37	32.58 $\pm$ 4.87	84.24 $\pm$ 0.92
CCVR [32]	59.82 $\pm$ 4.35	<b>79.83<math>\pm</math>1.66</b>	53.73 $\pm$ 8.48	80.43 $\pm$ 2.54	55.73 $\pm$ 4.00	81.83 $\pm$ 1.32	<b>55.00<math>\pm</math>2.29</b>	<b>85.61<math>\pm</math>1.12</b>
FEDROD [3]	<b>60.34<math>\pm</math>3.22</b>	77.10 $\pm$ 1.97	<b>56.74<math>\pm</math>5.59</b>	76.96 $\pm$ 4.26	<b>57.85<math>\pm</math>5.22</b>	81.08 $\pm$ 1.74	50.63 $\pm$ 9.49	84.75 $\pm$ 0.96
FEDNH [5]	39.14 $\pm$ 9.56	77.27 $\pm$ 1.53	45.13 $\pm$ 1.60	<b>81.84<math>\pm</math>0.88</b>	40.28 $\pm$ 2.78	<b>82.90<math>\pm</math>1.00</b>	39.18 $\pm$ 3.66	83.95 $\pm$ 1.23
<b>Our FEDETf</b>	<b>62.76<math>\pm</math>2.90</b>	<b>88.00<math>\pm</math>0.65</b>	<b>62.34<math>\pm</math>4.10</b>	<b>88.20<math>\pm</math>0.93</b>	<b>61.78<math>\pm</math>3.21</b>	<b>88.46<math>\pm</math>0.75</b>	<b>54.23<math>\pm</math>10.8</b>	<b>88.40<math>\pm</math>0.98</b>

ments, we evaluate the methods under strong Non-IID settings with  $\alpha \in \{0.1, 0.05\}$  [32, 52]. If without mentioning otherwise, the number of clients  $K = 20$  and we adopt full client participation. For CIFAR-10 and CIFAR-100 the number of local epochs  $E = 3$  and the number of communication rounds  $T = 200$ , while for Tiny-ImageNet, considering the high computation costs, we set  $E = 1$  and  $T = 50$ .

**Evaluation Metrics.** We test the generalization of the aggregated global model (General.) and the personalization of clients’ local models (Personal.). Generalization performance is validated on the balanced testset of each dataset after the global model is generated on the server. For each client, we split 70% of the local data for the trainset and 30% for the testset. Following [3], we validate the personalization performance on each client’s local testset after the local training and average the personalized accuracies.

**Implementation.** In all the experiments, we conduct three trials for each setting and present the mean accuracy and the standard deviation in the tables. For more implementation details, please refer to the Appendix.

## 5.2. Main Results

**Results under various vision datasets and data heterogeneity.** Table 1 shows the results of all methods on three vision datasets with Non-IID  $\alpha \in \{0.1, 0.05\}$ . *Our method achieves state-of-the-art performances in 11 out of 12 settings in both generalization and personalization. It is notable that except for our FEDETf, there is no comparable baseline that can achieve high results in all datasets.* Generally, our method has more significant improvement under more heterogeneous settings ( $\alpha = 0.05$ ), especially in CIFAR-10. We also visualize the learning curves in Figure 4. *Our FEDETf not only has higher accuracies but also*

*has faster and more steady convergence.*

For generalization, in most cases, FEDROD can improve the accuracies upon FEDAVG, which showcases the effectiveness of the balanced loss. However, the balanced loss cannot thoroughly solve the classifier bias problem, and *our method FEDETf which adopts the optimal classifier structure has large-margin gains over FEDROD.* We notice that the classifier retraining algorithm CCVR is not effective in all cases, which indicates that the retraining method is not practical enough for solving classifier biases.

For personalization, we find the personalized FL FEDREP and FEDROD are strong baselines. Compared with these personalized FL methods, our FEDETf also reaches the state-of-the-art in almost all cases. *The success of FEDETf in personalization is the result of training a better generalized global model and the effective local adaption of such a global model.*

### Results under different $K$ with partial client sampling.

We select the best baselines in Table 1 and conduct experiments on CIFAR-10 under various numbers of clients  $K$  with partial client sampling in Table 2. We set  $K \in \{50, 100\}$  and set the sampling rate as 0.4 and 0.6. To ensure fair comparisons, we randomly generate and save a sampling list in advance and let every method load the same sampling list during training. It is obvious that our FEDETf is the best method in both generalization and personalization. Compared with FEDAVG, the advantage of our method is more dominant under the smaller sampling rate, i.e. 0.4. With respect to generalization, when  $K = 50$ , the improvement is 5.2% for the sampling rate 0.4, and 3.8% for the sampling rate 0.6. *It indicates that the fixed ETF classifier is more robust than the learnable classifier to tackle system heterogeneity.* Moreover, our method has smaller variances

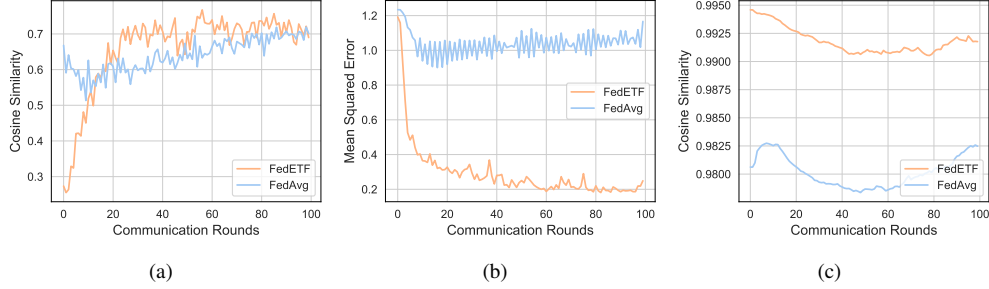


Figure 5. **Understanding feature alignment and neural collapse of FEDETf.** Test accuracy of final global models: FEDAVG 29.73%, FEDETf 54.95%. Experiments on CIFAR-10 with  $\alpha = 0.05$ . (a) Feature prototype consistency of clients' local models, higher values mean better feature alignment. (b) Neural collapse error of the aggregated global model, lower values mean greater neural collapse. (c) Local model consistency: averaged pair-wise cosine similarities of clients' local models, higher values mean smaller model drifts.

Table 4. **Evaluation (%) of FEDETf using different model architectures.** The dataset is CIFAR-10 with Non-IID  $\alpha = 0.1$ .

Methods	FEDAVG		FEDETf	
	General.	Personal.	General.	Personal.
DenseNet121	62.87 $\pm$ 2.23	85.66 $\pm$ 3.70	<b>74.92<math>\pm</math>2.75</b>	<b>91.83<math>\pm</math>0.67</b>
MobileNetV2	43.20 $\pm$ 4.45	87.07 $\pm$ 1.04	<b>57.43<math>\pm</math>12.0</b>	<b>89.88<math>\pm</math>0.40</b>
EfficientNet	35.92 $\pm$ 4.47	84.69 $\pm$ 1.26	<b>56.70<math>\pm</math>5.52</b>	<b>87.50<math>\pm</math>0.74</b>
ResNet20	52.76 $\pm$ 6.08	83.85 $\pm$ 0.89	<b>59.56<math>\pm</math>1.84</b>	<b>87.89<math>\pm</math>1.19</b>
ResNet32	53.22 $\pm$ 7.73	82.90 $\pm$ 4.31	<b>60.71<math>\pm</math>2.67</b>	<b>87.97<math>\pm</math>1.17</b>
ResNet56	57.09 $\pm$ 6.10	83.70 $\pm$ 4.65	<b>60.44<math>\pm</math>3.57</b>	<b>88.23<math>\pm</math>0.99</b>
WRN56_4	63.64 $\pm$ 4.91	86.76 $\pm$ 1.08	<b>66.30<math>\pm</math>3.88</b>	<b>89.94<math>\pm</math>0.52</b>

of accuracies, which also verifies its robustness.

**Results under different local epochs  $E$ .** In Table 3, generally, when the number of local epochs varies, our FEDETf also achieves constantly state-of-the-art performances. For FEDAVG and FEDROD, when  $E$  is larger, the generalization performance will decline. *However, for our FEDETf, the declines are weaker, showing its robustness and effectiveness. For personalization, FEDETf has more steady and promising results under different  $E$ .*

### 5.3. Understanding FEDETf

**Evaluation using different model architectures.** We consider two scenarios: *1) Different backbones.* DenseNet121 [13], MobileNetV2 [12, 37], and EfficientNet [39]. *2) Deeper and wider models.* Deeper models: ResNet20, ResNet32, and ResNet56 [23] (the larger number refers to the deeper model); wider models: ResNet56 and WRN56\_4 [23] (WRN: the abbreviation for Wide ResNet). The results are shown in Table 4. *Our method can also improve performance under various model architectures.* For models with different depths, we observe that FEDETf has larger superiority in shallower models. Specifically, FEDETf can release the full potential of ResNet20 (shallower and smaller model) to let it has even better performance than FEDAVG with ResNet56 (deeper and larger model). *It showcases the applicability of FEDETf that it can enable smaller models*

*to have better performances than the larger ones, saving both computation and communication costs in FL.*

**Why does FEDETf work well?** We explore how the features are learned in FEDETf compared with FEDAVG. We first examine the feature alignment of local models. In each round, after local training, we compute class prototypes (feature mean of each class) in each client and calculate the cosine similarities of clients' class-wise prototypes, which is analogous to NC1 in Section 3.2. Then we average all the cosine similarities to indicate the feature alignment, a larger value reveals more aligned clients' features. The results are in Figure 5 (a). *It shows that only after a few rounds, FEDETf has constantly stronger clients' feature alignment than FEDAVG, showing the fixed ETF classifier is effective to align local features.*

We also study whether FEDETf can help the global model reach neural collapse in terms of NC2 in Figure 5 (b). In each round, we first compute the class prototypes of the global model and calculate the pair-wise cosine similarities of these prototypes. In neural collapse optimality (Definition 3.1), the pair-wise cosine similarities of prototypes are  $-\frac{1}{K-1}$ . Hence, we calculate the mean square error between the global model's cosines and  $-\frac{1}{K-1}$  to indicate the neural collapse error. Results display that FEDETf has a much smaller neural collapse error than FEDAVG, and the error of FEDETf is decreasing along the training. *It indicates that FEDETf can help the global model reach neural collapse.* Note that FEDAVG has 29.73% in global test accuracy while FEDETf has 54.95%. *It also verifies that the global model's generalization is connected with neural collapse optimality in FL, which is consistent with the observations in centralized training [26].*

Additionally, we study why FEDETf works well from the perspective of model drift (local model consistency) in Figure 5 (c). Results demonstrate that FEDETf has stably higher local model consistency than FEDAVG throughout the training, indicating lower model drift. It is consistent with the intuition that the fixed ETF classifier can reduce



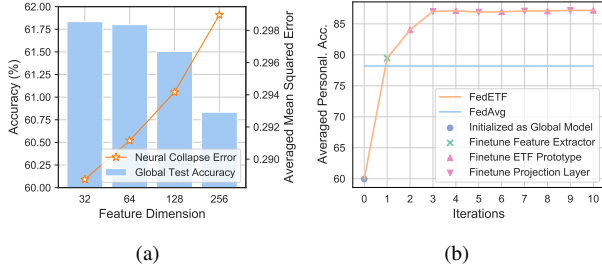


Figure 6. **Understanding feature dimension and local personalization in FEDETf.** Experiments on CIFAR-10 with  $\alpha = 0.1$ . (a) How feature dimension affects FEDETf’s generalization and neural collapse. (b) How personalization is reached in each iteration of FEDETf’s local finetuning.

the model drifts of feature extractors, resulting in better aggregation and convergence.

**How feature dimension affects FEDETf.** We analyse how the feature dimension  $d$  of ETF affects FEDETf’s performance on CIFAR-10 in Figure 6 (a). We find that smaller  $d$  will cause smaller neural collapse errors and are slightly beneficial to generalization. Random high-dimensional vectors are more prone to be orthogonal, so we suppose that prototypes in higher dimensions are more likely to be orthogonal. In CIFAR-10, the number of classes  $K = 10$ , and the ETF angles are obtuse with  $-0.11$  pair-wise cosines. Therefore, it is hard for high-dimensional features to collapse into the obtuse angle’s structure. *We suggest setting the feature dimension  $d$  in FEDETf according to the number of classes  $C$ .* If  $C$  is small, it also requires a relatively small  $d$  to improve neural collapse.

**How personalization is reached during local finetuning.** We visualize the averaged personalized accuracies of different iterations during FEDETf’s local finetuning in Figure 6 (b). At first, when clients’ local models are initialized as the final global model, the personalization is poor. After finetuning the feature extractor, FEDETf has better results than the baseline FEDAVG with finetuning. Then alternatively finetuning the ETF classifier and projection layer further improves the personalization and makes the accuracy converge to a higher point.

#### 5.4. Ablation Study

We conduct the ablation study of FEDETf in terms of generalization<sup>5</sup> on CIFAR-10 with Non-IID  $\alpha \in \{0.1, 0.05\}$ . *It is found that every module in FEDETf plays an important role and the modules strengthen each other to realize better performances.* If taking one module off, the performance will meet severe declines, but the results are still better than FEDAVG in general. We notice the balanced loss module is more important under a more heterogeneous

<sup>5</sup>Figure 6 (b) can be viewed as the ablation study of the personalized local finetuning stage.

Table 5. **Ablation study of FEDETf in terms of global model’s generalization.** The dataset is CIFAR-10.

Methods/NonIID( $\alpha$ )	0.1	0.05
FEDAVG	43.75 $\pm$ 0.42	38.47 $\pm$ 1.95
Ours w/o Projection Layer	44.92 $\pm$ 6.22	41.91 $\pm$ 1.47
Ours w/o Balanced Loss	46.06 $\pm$ 0.75	37.63 $\pm$ 4.45
Ours w/o Learnable Temperature	49.80 $\pm$ 4.52	46.07 $\pm$ 1.61
<b>Ours</b>	<b>56.46<math>\pm</math>4.18</b>	<b>53.98<math>\pm</math>1.29</b>

environment, and this observation is consistent with previous works in neural collapse [44] and FL [3]. It is also notable to emphasize the significance and necessity of the projection layer. Our method without a projection layer only has marginal gains over FEDAVG. We also find that FEDNH has relatively poor performances in Table 1, especially on CIFAR-100 and Tiny-ImageNet, and we suppose the main cause may be that FEDNH does not adopt a projection layer to map the raw features into a space where neural collapse is more prone to happen. Moreover, the learnable temperature is also crucial for FEDETf to adaptively adjust the softmax temperature so as to meet the learning dynamics of feature representations.

## 6. Conclusion

In this paper, we fundamentally solve the classifier biases caused by data heterogeneity in FL by proposing a neural-collapse-inspired solution. Specifically, we employ a simplex ETF as a fixed classifier for all clients during federated training, which allows them to learn unified and optimal feature representations. Afterward, we introduce a novel finetuning strategy to enable clients to have more personalized local models. Our method achieves the state-of-the-art performance regarding both generalization and personalization compared to strong baselines, as shown by extensive experimental results on CIFAR-10/100 and Tiny-ImageNet. Furthermore, we gained insights into understanding the effectiveness and applicability of our approach.

## Acknowledgments

This work was supported by the National Key Research and Development Project of China (2021YFC3340300), National Natural Science Foundation of China (U19B2042), the Zhejiang Provincial Key Research and Development Project (2022C03106), University Synergy Innovation Program of Anhui Province (GXXT-2021-004), Academy Of Social Governance Zhejiang University, Fundamental Research Funds for the Central Universities (226-2022-00064). This work was supported in part by the National Key R&D Program of China (Project No. 2022ZD0115100), the Research Center for Industries of the Future (RCIF) at Westlake University, and Westlake Education Foundation.

## References

- [1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2020.
- [2] Mohammed Adnan, Shivam Kalra, Jesse C Cresswell, Graham W Taylor, and Hamid R Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1):1953, 2022.
- [3] Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.
- [4] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pages 2089–2099. PMLR, 2021.
- [5] Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. Tackling data heterogeneity in federated learning with class prototypes. 2023.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [7] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [8] Pengfei Guo, Dong Yang, Ali Hatamizadeh, An Xu, Ziyue Xu, Wenqi Li, Can Zhao, Daguang Xu, Stephanie Harmon, Evrim Turkbey, et al. Auto-fedrl: Federated hyperparameter optimization for multi-institutional medical image segmentation. *arXiv preprint arXiv:2203.06338*, 2022.
- [9] Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.
- [12] Andrew Howard, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Menglong Zhu. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. 2018.
- [13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [14] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *CVPR*, 2022.
- [15] Wenke Huang, Mang Ye, Zekun Shi, He Li, and Bo Du. Rethinking federated learning with domain shift: A prototype view. In *CVPR*, 2023.
- [16] Wenlong Ji, Yiping Lu, Yiliang Zhang, Zhun Deng, and Weijie J Su. An unconstrained layer-peeled perspective on neural collapse. *arXiv preprint arXiv:2110.02796*, 2021.
- [17] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.
- [18] Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021.
- [19] Vignesh Kothapalli, Ebrahim Rasromani, and Vasudev Awatramani. Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*, 2022.
- [20] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.
- [21] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.
- [22] Bo Li, Mikkel N Schmidt, Tommy S Alstrøm, and Sebastian U Stich. Partial variance reduction improves non-convex federated learning on heterogeneous data. *arXiv preprint arXiv:2212.02191*, 2022.
- [23] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.
- [24] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Confer-*

- ence on Machine Learning, pages 6357–6368. PMLR, 2021.
- [25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.
- [26] Xiao Li, Sheng Liu, Jinxin Zhou, Xinyu Lu, Carlos Fernandez-Granda, Zhihui Zhu, and Qing Qu. Principled and efficient transfer learning of deep models via neural collapse. *arXiv preprint arXiv:2212.12206*, 2022.
- [27] Zexi Li, Qunwei Li, Yi Zhou, Wenliang Zhong, Guan-nan Zhang, and Chao Wu. Edge-cloud collaborative learning with federated and centralized features. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’23, page 1949–1953, New York, NY, USA, 2023. Association for Computing Machinery.
- [28] Zexi Li, Tao Lin, Xinyi Shang, and Chao Wu. Revisiting weighted aggregation in federated learning with neural networks. In *International Conference on Machine Learning*. PMLR, 2023.
- [29] Zexi Li, Jiaxun Lu, Shuang Luo, Didi Zhu, Yunfeng Shao, Yinchuan Li, Zhimeng Zhang, Yongheng Wang, and Chao Wu. Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching. *IEEE Transactions on Big Data*, 2022.
- [30] Zexi Li, Feng Mao, and Chao Wu. Can we share models if sharing data is not an option? *Patterns*, 3(11):100603, 2022.
- [31] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.
- [32] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.
- [33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [34] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.
- [35] Vardan Pappayan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. volume 117, pages 24652–24663. National Acad Sciences, 2020.
- [36] Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020.
- [37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [38] Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. pages 2218–2224, 2022.
- [39] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
- [40] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022.
- [41] Christos Thrampoulidis, Ganesh R Kini, Vala Vakilian, and Tina Behnia. Imbalance trouble: Revisiting neural-collapse geometry. *arXiv preprint arXiv:2208.05512*, 2022.
- [42] Tom Tirer and Joan Bruna. Extended unconstrained features model for exploring deep neural collapse. In *International Conference on Machine Learning*, pages 21478–21505. PMLR, 2022.
- [43] Liang Xie, Yibo Yang, Deng Cai, and Xiaofei He. Neural collapse inspired attraction-repulsion-balanced loss for imbalanced learning. *Neurocomputing*, 2023.
- [44] Yibo Yang, Liang Xie, Shixiang Chen, Xiangtai Li, Zhouchen Lin, and Dacheng Tao. Do we really need a learnable classifier at the end of deep neural network? volume 33, pages 2351–2363, 2022.
- [45] Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning. In *International Conference on Learning Representations*, 2023.

- [46] Rui Ye, Zhenyang Ni, Chenxin Xu, Jianyu Wang, Siheng Chen, and Yonina C Eldar. Fedfm: Anchor-based feature matching for data heterogeneity in federated learning. *arXiv preprint arXiv:2210.07615*, 2022.
- [47] Rui Ye, Mingkai Xu, Jianyu Wang, Chenxin Xu, Siheng Chen, and Yanfeng Wang. Feddisco: Federated learning with discrepancy-aware collaboration. *arXiv preprint arXiv:2305.19229*, 2023.
- [48] Honglin Yuan, Warren Richard Morningstar, Lin Ning, and Karan Singhal. What do we mean by generalization in federated learning? In *International Conference on Learning Representations*, 2021.
- [49] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 35:21414–21428, 2022.
- [50] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedala: Adaptive local aggregation for personalized federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11237–11244, 2023.
- [51] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedcp: Separating feature information for personalized federated learning via conditional policy. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 3249–3261, New York, NY, USA, 2023. Association for Computing Machinery.
- [52] Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning*, pages 26311–26329. PMLR, 2022.
- [53] Tailin Zhou, Jun Zhang, and Danny Tsang. Fedfa: Federated learning with feature anchors to align feature and classifier for heterogeneous data. *arXiv preprint arXiv:2211.09299*, 2022.
- [54] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. volume 34, pages 29820–29834, 2021.

## Appendix

### A. Pseudo Codes of the Proposed Method

To better present our approach and demonstrate the workflow, we give the pseudo codes of FEDETF. The pseudo code of the federated learning (FL) training is shown in Algorithm 1, and the pseudo code of the personalized local finetuning is shown in Algorithm 2.

---

#### Algorithm 1: FEDETF FL Training

---

**Input:** Clients  $\{1, \dots, K\}$ , communication round  $T$ , local epoch  $E$ , initial model  $\mathbf{w}^1 = \{\mathbf{u}, \mathbf{p}, \beta\}$ , feature dimension  $d$ , balanced loss hyperparameter  $\gamma$ .

**Output:** Gloabl model  $\mathbf{w}^g$ .

Synthesize a simplex ETF  $\mathbf{V}_{ETF} \in \mathbb{R}^{d \times C}$  by Eq.

(3) as the fixed classifier for all clients;

**for**  $t = 1, \dots, T$  **do**

**for** client  $k = 1, \dots, K$  **in parallel do**

$\mathbf{w}_k^t \leftarrow \mathbf{w}^t$ ;

**for** local epoch  $e = 1, \dots, E$  **do**

            Obtain  $\mathcal{L}_k^g$  by Eq. (6, 7, 8);

$\mathbf{w}_k^t \leftarrow \mathbf{w}_k^t - \eta \nabla \mathcal{L}_k^g(\mathbf{w}_k^t)$ ;

**end**

**end**

    The server updates  $\mathbf{w}^{t+1}$  by Eq. (2).

**end**

The final global model  $\mathbf{w}^g = \mathbf{w}^T$ .

---

### B. Implementation Details

**Models and Data.** Our model implementations of the ResNet series and the DenseNet are referred from the codes of [23]. The model implementation of the EfficientNet is referred from the official code in [39], and the implementation of the MobileNetv2 is referred from [37, 12]. For the data, we use the Dirichlet-sampling-based data partition adopted in [31, 3, 5, 32]. It considers a class-imbalanced data heterogeneity, controlled by hyperparameter  $\alpha$ , and smaller  $\alpha$  refers to more Non-IID data of clients. When  $\alpha < 1$ , the data are considered to be rather Non-IID, which means that most of the training samples of one class are likely assigned to a small portion of clients [3]. In our Dirichlet implementation, when  $\alpha$  goes smaller, the number of samples in each client along with the class distribution of each client both become more heterogeneous, which is realistic in practical scenarios. We use the same Tiny-ImageNet dataset as in [5].

**Local learning rate and optimizer.** For CIFAR-10 the local learning rate (LR)  $\eta = 0.04$ , and for CIFAR-10 and Tiny-ImageNet,  $\eta = 0.01$ . For clients, we use SGD optimizer with momentum 0.9 and weight decay  $5 \times 10^{-4}$ .

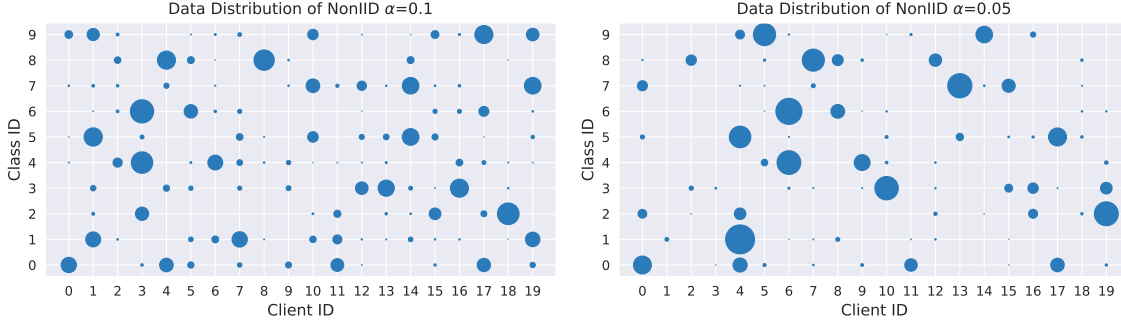


Figure 7. **Visualization of clients' data distributions.** Random seed is 8. Left: data distributions of Non-IID  $\alpha = 0.1$  with 20 clients. Right: data distributions of Non-IID  $\alpha = 0.05$  with 20 clients.

---

**Algorithm 2:** FEDETF Personalized Finetuning

---

**Input:** Clients  $\{1, \dots, K\}$ , iteration round  $T_p$ ,  
epoch for each stage  $E$ , final global model  
 $\mathbf{w}^g = \{\mathbf{u}, \mathbf{p}, \beta, \mathbf{V}_{ETF}\}$ .

**Output:** Personalized local models  $\{\mathbf{w}_k^p\}_{k=1}^K$ .  
Assign the final global model  $\mathbf{w}^g$  as clients' initial  
local models.

*Finetune the feature extractor.*

**for** client  $k = 1, \dots, K$  **in parallel do**  
     $\hat{\mathbf{w}}_k = \{\mathbf{u}, \beta\}$ ,  $\bar{\mathbf{w}}_k = \{\mathbf{p}, \mathbf{V}_{ETF}\}$ ;  
    **for** local epoch  $e = 1, \dots, E$  **do**  
        Obtain  $\mathcal{L}_k^p$  by Eq. (9, 10, 11);  
         $\hat{\mathbf{w}}_k \leftarrow \hat{\mathbf{w}}_k - \eta \nabla \mathcal{L}_k^p(\hat{\mathbf{w}}_k)$ ;  
    **end**

**end**

**for**  $t = 1, \dots, T_p$  **do**

**for** client  $k = 1, \dots, K$  **in parallel do**  
        *Finetune the ETF classifier.*

$\hat{\mathbf{w}}_k = \{\mathbf{V}_{ETF}, \beta\}$ ,  $\bar{\mathbf{w}}_k = \{\mathbf{p}, \mathbf{u}\}$ ;  
        **for** local epoch  $e = 1, \dots, E$  **do**  
            Obtain  $\mathcal{L}_k^p$  by Eq. (9, 10, 11);  
             $\hat{\mathbf{w}}_k \leftarrow \hat{\mathbf{w}}_k - \eta \nabla \mathcal{L}_k^p(\hat{\mathbf{w}}_k)$ ;  
        **end**

*Finetune the projection layer.*

$\hat{\mathbf{w}}_k = \{\mathbf{p}, \beta\}$ ,  $\bar{\mathbf{w}}_k = \{\mathbf{V}_{ETF}, \mathbf{u}\}$ ;  
        **for** local epoch  $e = 1, \dots, E$  **do**  
            Obtain  $\mathcal{L}_k^p$  by Eq. (9, 10, 11);  
             $\hat{\mathbf{w}}_k \leftarrow \hat{\mathbf{w}}_k - \eta \nabla \mathcal{L}_k^p(\hat{\mathbf{w}}_k)$ ;  
        **end**

**end**

**end**

The personalized local models are

$$\{\mathbf{w}_k^p = \hat{\mathbf{w}}_k \cup \bar{\mathbf{w}}_k\}_{k=1}^K.$$


---

in their official implementations or papers. For DITTO, the learning setting of the personalized model is the same as the one of the global model. For FEDREP, the epoch number of the classifier training and the epoch number of the feature extractor training are the same and are set as  $E$ . For FEDROD, we set  $\gamma = 1$ . For CCVR, the number of virtual features is 10 per class, and the number of classifier calibration training epochs is 100. For FEDNH, the smoothing hyperparameter  $\rho = 0.9$  as suggested in the paper [5].

**Randomness.** We set the same random seeds for all methods in the same setting. The random seed list is  $\{7, 8, 9, 10\}$ . For the extremely Non-IID settings when  $\alpha = 0.05$ , we use the random seeds that can ensure all clients can be assigned a proportion of training data (on the contrary, some random seeds will generate a data partition where particular clients have zero data samples).

**Environments.** All experiments are conducted in PyTorch with Quadro RTX 8000 GPUs.

## C. Visualization

### C.1. Visualization of clients' data distributions.

Here, we additionally visualize the clients' data distributions mainly adopted in the main paper. In the main paper, we adopt  $\alpha \in \{0.1, 0.05\}$  with 20 clients in Tables 1, 3, and 4. We visualize the data distributions in Figure A. It shows that in both settings, the clients have extremely heterogeneous data distributions. Especially when  $\alpha = 0.05$ , all clients have some classes missing, and some clients have extremely rare data (e.g. client 1). We note that these settings are very realistic in practical FL scenarios.

Following [7], we adopt a learning rate decaying scheduler, which decays the local LR by 0.99 in each round.

**Hyperparameters.** For FEDETF, we set the feature dimension to the number of classes, i.e.  $d = C$ ; the initial temperature  $\beta = 1$ ;  $\gamma = 1$ . We set  $\mu_{FedProx} = 0.001$  in FEDPROX and  $\alpha_{FedDyn} = 0.01$  in FEDDYN as suggested