所有client基于hidden共同计算得到一个共同的分类层，使得分类层的输出在一个球面上，使得分类更加容易。分类层固定不变，特征提取层采用FedAvg的方式训练。其中，分类层可以视作所有client计算得到的分类层的累加

SphereFed: Hyperspherical Federated Learning

Xin Dong[1(✉)], Sai Qian Zhang[1], Ang Li[2], and H.T. Kung[1]

[1] Harvard University, Cambridge, USA
xindong@g.harvard.edu
[2] UT Dallas, Richardson, USA

Abstract. Federated Learning aims at training a global model from multiple decentralized devices (*i.e.* clients) without exchanging their private local data. A key challenge is the handling of non-*i.i.d.* (independent identically distributed) data across multiple clients that may induce disparities of their local features. We introduce the Hyperspherical Federated Learning (SphereFed) framework to address the non-*i.i.d.* issue by constraining learned representations of data points to be on a unit hypersphere shared by clients. Specifically, all clients learn their local representations by minimizing the loss with respect to a fixed classifier whose weights span the unit hypersphere. After federated training in improving the global model, this classifier is further calibrated with a closed-form solution by minimizing a mean squared loss. We show that the calibration solution can be computed efficiently and distributedly without direct access of local data. Extensive experiments indicate that our SphereFed approach is able to improve the accuracy of multiple existing federated learning algorithms by a considerable margin (up to 6% on challenging datasets) with enhanced computation and communication efficiency across datasets and model architectures.

Keywords: Federated learning · Efficient classifier calibration

1 Introduction

Federated learning (FL) is an emerging machine learning paradigm in which distributed clients learn on private data and communicate with a coordinating server to train a single global model that generalizes well across local data [51,69]. One of its major challenges is the handling of non-*i.i.d.* (independent identically distributed) local data across clients [32,40,42]. Non-*i.i.d.* local data leads to disparity of local models after learning on private data [94]. For instance, different feature[1] extractors in local models may learn biased and discrepant input-to-feature mapping functions for the same class [18,41,89]. This obstructs the convergence of collaborative training.

[1] The terms representation and feature are used interchangeably.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19809-0_10.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13686, pp. 165–184, 2022.
https://doi.org/10.1007/978-3-031-19809-0_10

Existing federated learning algorithms primarily tackle the non-*i.i.d.* problem in two phases: (i) *In the local learning phase*, regularization terms [2,43,80] and additive objectives [41,84,97] are used to control distances among local models by constraining the learning process. (ii) *In the post-learning phase*, the inevitable divergence of local models is corrected with additional information exchange [24,32,66,79,82] and advanced aggregation strategies such as normalized averaging [76], distillation [39], and so on [3,5,60,87].

In this work, we argue that federated learning can also be improved *in the pre-learning phase*; this is a novel research direction complementary to existing approaches. A key insight is the use of a fixed classifier (*e.g.*, the last fully-connected layer) that serves as a template of the feature extractor's output for all clients. Note the loss function is often computed on the inner product between the feature vector (*i.e.*, output of the feature extractor) and the classifier's weight vectors. During local training, the feature extractor is optimized to project data from the i-th class to feature vectors that have the maximum inner product with the i-th of row of the classifier. We refer to the classifier as learning target of the feature extractor. However, higher data heterogeneity leads to a larger disparity of classifiers (in terms of both norms and directions) across clients. In this regard, if the local classifiers can be aligned, clients would have more consistent learning targets without modifying the learning procedure. Unfortunately, a real-time classifier synchronization carries prohibitively high communication overhead for federated learning. To avoid this communication cost, we use instead, for all clients, a fixed classifier constructed from orthonormal basis vectors.

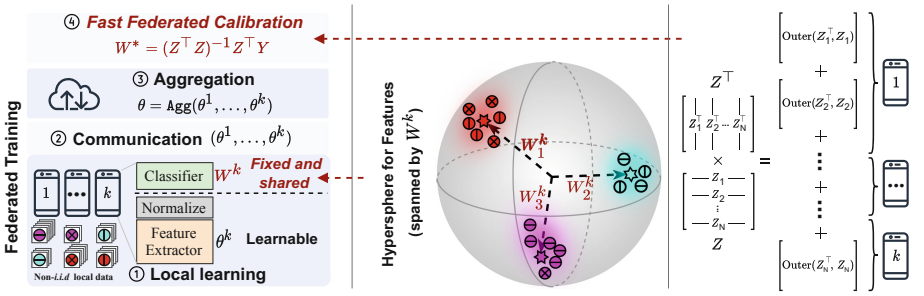

Fig. 1. *Left*: An overview of SphereFed (Hyperspherical Federated Learning). Before federated training starts, we construct a fixed shared classifier whose weight vectors span an unit hypersphere. After federated training ends, we calibrate the classifier in a distributed manner. *Middle*: All clients share the same hypersphere and learn to map local samples (markers represent clients) from the same class (colors represent classes) to the same area on the hypersphere whose centroid (the pentagram) corresponds to a weight vector. *Right*: Leveraging the linearity of the classifier, we derive its closed-form optimum which can be precisely computed by distributed clients. $Z^\top Z$ and $Z^\top Y$ are computed distributively because a matrix multiplication can be implemented as an accumulation of outer products and each outer product depends only on one client. (Color figure online)

Motivated by this insight, we propose to construct a classifier whose weight vectors span an unit hypersphere *before the federated training starts*. Throughout federated training, this classifier is fixed and shared by all clients. Meanwhile, we also normalize the feature representation to the same hypersphere. During local learning, clients' feature extractors learn to map data samples from the *same* class to the *same* area on the hypersphere whose centriod is the corresponding row vector of the classifier. As a result, the local features learned by different clients for data belonging to the same class are better aligned and the interference among local models are reduced, leading to an improved accuracy of the global model.

We name our approach *Hyperspherical Federated Learning* (SphereFed), which is a generic framework compatible with existing federated learning algorithms. An overview of the framework is illustrated in Fig. 1. SphereFed does **not** introduce extra hyper-parameters nor requiring additional computation. In fact, a pre-defined classifier eliminates the need of communication and brings improved efficiency to the system. Given that the classifiers are frozen during federated training, we propose to calibrate the classifier after federated training to achieve its optimum in a provable and lightweight manner. We first derive the closed-form optimum of the classifier leveraging its linearity and find that this closed-form solution can be precisely computed in a distributed manner without direct access to the private features (or data). We name this calibration method *Fast Federated Calibration* (FFC), which is provable and efficient compared with state-of-the-art methods (*e.g.*, [48]) that depend on synthetically generated virtual features.

We conduct extensive experiments to demonstrate that the proposed SphereFed method is compatible with and complementary to several existing federated learning algorithms, capable of introducing up to 6% improvement on testing accuracy. Further experiments show that our proposed calibration achieves a performance gain comparable to the oracle fine-tuning with real features, verifying its theoretical optimality. A set of ablation studies are further presented to understand the efficacy of each design component in SphereFed.

2 Related Work

Standard Federated Learning. Federated learning (FL) was originally proposed by [51,69]. To address the non-*i.i.d.* problem, works have been pursued in two directions: imposing additional constraints in the local learning phase [64,66] and conducting weight correction in the post learning phase [48,60,76,87]. There have also been studies that tackle the non-*i.i.d.* issue by augmenting the on-client and on-server data with public [39] or synthetic [85,97] samples.

Personalized Federated Learning. Personalized federated learning (pFL) differs from FL, by relaxing the setting of standard FL to allow each client to have its personalized local model via, e.g., additional local epochs after standard FL [11,44]. In general, a personalized local model is more likely to obtain

better accuracy on a local test set than the single global model, but the personalized local model could be more biased and less general to data from other sources [29,77]. So, pFL and FL has different use focuses and application scenarios. Inspired from transfer learning [57,86], a line of pFL methods learns local private parameters for the classifier but uses a shared feature extractor [7,15,70]. A concurrent and most related work is FedBABU [54] which finds that fixing the classifier during collaborative learning is beneficial to the personalization process. Although this finding is consistent to our observations, our contribution is substantially different from FedBABU. First, we focus on FL while FedBABU focuses on pFL. Second, we ensure a stable performance gain resulting from an in-depth analysis on the benefit of fixing classifier. Third, we further propose a provable calibration method to improve the classifier after federated training.

Decoupling Layers for Federated Learning. Dealing layers at different depths with varying strategies has demonstrated effectiveness for many tasks in centralized training [23], like few-shot learning [65,83], domain adaption [28,75] and meta-learning [55,59]. Such layer decoupling studies could also benefit federated learning applications. For instance, parameters from different layers can be updated and synchronized with different frequency to save communication cost [12,13,16]. FedRecon [68] splits a model into global/local parts and reconstructs the local part on clients in each round to improve privacy and reduce communication. FedUFO [89] resorts an adversary module to reduce the divergence of feature extractors on clients. A most related work is CCVR [48] which also focuses on the classifier. CCVR conducts on-server calibration for the classifier by fine-tuning it with virtual features sampled from Gaussian distributions. This work uses fundamentally different methodologies for the classifier (closed-form classifier fine tuning), and has higher performance gains and less communication/computation costs against CCVR.

Hyperspherical Representation. To the best of our knowledge, this is the first work introducing hyperspherical representation to address the non-*i.i.d.* challenge in FL. This combination is not trivial but motivated by analytical justifications and empirical supports as elaborated in the remaining sections. Hyperspherical representation has been widely adapted by studies on face recognition [45,95], long-tail recognition [31], regression [52], metric learning [88,96] and contrastive learning [33] to enhance the discriminative power of features. In this work, under the context of FL, we use hyperspherical features with fixed targets to align the learning objectives and minimize cross-party interference.

3 Federated Learning with Non-*i.i.d.* Clients

3.1 Terminologies

We consider K clients and a central server in a federated learning system. Each client $k \in [K]$ has a local and private dataset $\mathcal{D}^k$. We focus on the non-*i.i.d.* data

setting where local datasets could have heterogeneous distributions [40]. The goal is to train a single global classification model collaboratively which performs well on the global test set. The loss function is represented using $\mathcal{L}(\cdot, \cdot)$.

For a single training example $(\mathbf{x}, y)$, let $\mathbf{z} = f_\theta(\mathbf{x}) \in \mathbb{R}^l$ denote the l-dimension feature vector given a feature extractor $f_\theta(\cdot)$ parameterized by θ. The classifier $h_\mathbf{W}(\cdot)$ takes $\mathbf{z}$ as input and makes the final prediction after a linear transformation $\mathrm{o} = h_\mathbf{W}(\mathbf{z}) = \mathbf{W}\mathbf{z} + \mathbf{b}$ with a weight matrix $\mathbf{W} \in \mathbb{R}^{C \times l}$, where C is the number of classes. For simplicity, we omit the bias term $\mathbf{b}$ in future equations.

同类别不同client的分类层权重，non-IID程度越高，分类层的权重越发散

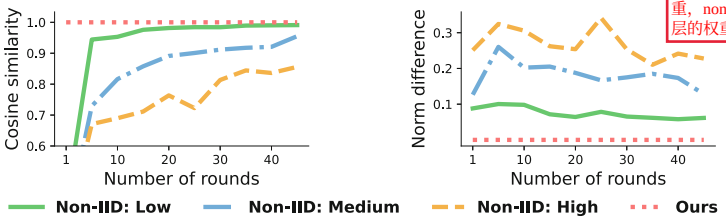

Fig. 2. Direction and norm alignment of classifiers' weights across clients. For the sake of simplicity, we train ResNet18 on CIFAR-100 with 10 clients for 50 rounds, using FedAvg in this empirical study. The non-*i.i.d.* level is controlled by the concentration parameter of a Dirichlet distribution [32,40,42]. There is a clear negative correlation between non-*i.i.d.* level and consistency of classifiers' weights across clients, which incurs inconsistent local learning targets.

3.2 Non-*i.i.d.* Data Leads to Inconsistent Local Learning Targets

In each round of standard federated learning, each client optimizes the feature extractor and the classifier $(\theta, \mathbf{W})$ jointly. Then each client sends its updated feature extractor and classifier $(\theta^k, \mathbf{W}^k)$ to the central server which aggregates (*e.g.*, averages [51]) all received local models into a single global one used for the next round. Prior studies [40] focus on either the local training loss function or an advanced aggregation strategy.

In this work, we pay special attention to the classifier. A classifier is the closet layer to the loss function and the i-th row of its weights $\mathbf{w}_i^k$ acts as a feature template and the learning target of the i-th class for the feature extractor. As a result, the disparity of classifiers across clients induces inconsistent local learning targets and further engenders local feature extractors' disparity. A performance degradation may occur after aggregation in the central server.

The above hypothesis is verified by empirical observations. To show that, we rewrite the output of the classifier as,

将每一个类的classifier拆分

$$\mathbf{W}^k\mathbf{z} = \left[\mathbf{w}_1^k\mathbf{z}, \ldots, \mathbf{w}_i^k\mathbf{z}, \ldots, \mathbf{w}_C^k\mathbf{z}\right], \text{ where } \mathbf{w}_i^k\mathbf{z} = \|\mathbf{w}_i^k\|\|\mathbf{z}\| \cdot \cos\left(\angle\left(\mathbf{w}_i^k, \mathbf{z}\right)\right). \quad (1)$$

$\angle(\cdot, \cdot)$ denotes the angle between two vectors and $\|\cdot\|$ is the euclidean norm of a vector. A local feature extractor f_θ^k learns to maximize the output of the ground-truth class, $\mathbf{w}_i^k\mathbf{z}$, $\forall i = y$, and minimize outputs of other classes $\mathbf{w}_j^k\mathbf{z}$, $\forall j \neq y$. In

summary, Eq. (1) highlights that both norms and directions of classifier's weight vectors impact the optimization of feature extractor and thus (at least partially) effect the distribution of features generated by the feature extractor.

We empirically find that there is a clear negative correlation between non-*i.i.d.* degree and the consistency of classifiers' weights (in terms of both norm and direction) across clients. We compute the cosine similarity (Eq. (2) and Fig. 2, *Left*) and norm difference (Eq. (3) and Fig. 2, *Right*) of classifier weight vectors for the same class but from arbitrary two different clients ($k_1 \neq k_2$ and $1 \leq k_1, k_2 \leq K$) using FedAvg [51].

$$\mathbb{E}_{c\sim[C],k_1\neq k_2}\left[\frac{\mathbf{w}_c^{k_1}\cdot\mathbf{w}_c^{k_2}}{\|\mathbf{w}_c^{k_1}\|\|\mathbf{w}_c^{k_2}\|}\right] \quad (2) \qquad \mathbb{E}_{c\sim[C],k_1\neq k_2}\left[\left|\|\mathbf{w}_c^{k_1}\| - \|\mathbf{w}_c^{k_2}\|\right|\right] \quad (3)$$

According to Fig. 2, data with a higher degree of non-*i.i.d.* is associated with less direction alignment and larger magnitude difference of classifier weight vectors among clients, which lead to weaker consistency of the local learning targets. This will further engender non-overlapped feature distributions for the same class on different clients as illustrated in Fig. 3 (*Left*).

4 Hyperspherical Federated Learning

4.1 Hyperspherical Representation

A simple yet effective tweak to bypass the aforementioned issue is to align features from clients on a hypersphere, aided by a fixed classifier. In Eq. (1), we show that both the norm and the direction of k-th weight vector $\mathbf{w}_i^k$ play crucial roles in the learning of f_{θ^k} but a non-*i.i.d.* data distribution causes a disorder of norms and directions. To tackle this problem, we consider to construct $\mathbf{W}^k = \{\mathbf{w}_i^k\}_{i=1}^C$ manually, which has a unit norm and orthogonal components,

$$\mathbf{W}^k = \left\{\mathbf{w}_i^k\right\}_{i=1}^C, \quad \text{where} \quad \|\mathbf{w}_i^k\| = 1 \text{ and } \mathbf{w}_i^k \perp \mathbf{w}_j^k, \ \forall\, i = j. \quad (4)$$

Note that $\{\mathbf{w}_i^k\}_{i=1}^C$ span an l-dimension unit hypersphere. The orthogonality among $\{\mathbf{w}_i^k\}_{i=1}^C$ ensures the maximum separation between arbitrary pair of classes. In addition, the uniformly unit norm guarantees balance in classes. Feature normalization is further adapted to project feature vectors on the same unit hypersphere, $\tilde{\mathbf{z}} = \mathbf{z}/\|\mathbf{z}\|$. Normalizing features enables f_θ^k to focus on learning feature vectors' directions and makes federated training process more robust to feature magnitude. Given a data point $(\mathbf{x}, y)$, a feature extractor f_{θ^k} maps the input $\mathbf{x}$ to a feature vector $f_{\theta^k}(\mathbf{x})/\|f_{\theta^k}(\mathbf{x})\|$ on the unit hypersphere, using the corresponding weights $\mathbf{w}_y^k$ as the target of mapping.

To ensure that all clients have the same learning targets (*i.e.*, the same classifier $h_{\mathbf{W}^k}$), we share the constructed $\mathbf{W}^k$ with all clients and keep the shared $\mathbf{W}^k$ fixed throughout the federated training process. By doing this, local classifiers become consistent automatically without costly frequent inter-client synchronization. Since all clients now share the same learning targets, different local feature extractors on clients learn to map local data samples from the i-th class

to the same area on the hypersphere with $\mathbf{w}_i^k$ as the centroid of that area. As a result, the norms and directions of local features are aligned with reduced interference across clients. In addition, the features from different classes have minimized overlaps and balanced magnitudes. An illustration of hyperspherical features can be found in Fig. 1 (Middle).

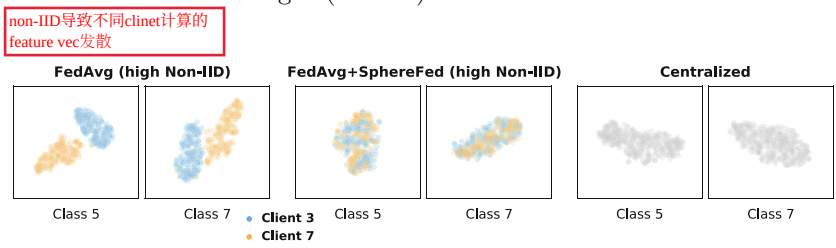

Fig. 3. A qualitative study of Hyperspherical Federated Learning (SphereFed). In the left and middle sub-figures, dots in different colors represent features of the same class but generated by local models f_{θ^k} of different clients. SphereFed encourages consistency among clients' features by aligning local learning targets (Color figure online).

The benefits of adapting the proposed hyperspherical features (and each design component described above) are revealed in a qualitative evaluation (Fig. 3). For more detailed ablation study including quantitative results, please refer to Sect. 5.3. In Fig. 3, we plot a certain class's features from different clients to visualize local features alignment across clients. Three kinds of methods are compared including FedAvg [51] (with conventional classifier), standard centralized training, and SphereFed. We use MobileNetV2 for all three methods. For FedAvg and SphereFed, we partition the CIFAR-100 dataset to 10 clients according to the Dirichlet distribution with the concentration parameter α set as 0.1 to simulate the high non-*i.i.d.* scenario. For the sake of visualization, we randomly select two clients (*e.g.*, the 3-rd and 7-th clients in Fig. 3) and use their local models (at the 90-th round) to generate features of two random classes (*e.g.*, the class 5 and 7 in Fig. 3). For centralized training, we train MobileNetV2 for 120 epochs to convergence and use the learnt model to generate features. The dimension of raw features is 1280 and we use t-SNE [49] to reduce the dimension to 2 for visualization. According to Fig. 3, in FedAvg, local models learn divergent mapping functions and thus features from the same class but different clients are biased to different distributions (*i.e.*, non-overlapped clusters in Fig. 3). While, our hyperspherical features aligns well across clients as the centralized training (*i.e.*, fused clusters in Fig. 3).

4.2 Using Mean Squared Error Loss on Hyperspherical Features

A widely accepted training method for classification tasks is to apply the softmax function [6] to the classifier output $\mathbf{W}^k\mathbf{z}$ before calculating the cross entropy (CE) loss. Combination of CE and softmax function is known to be sensitive to the scale of its input [23,74]. However, in SphereFed, the classifier's output, has

less-than-one scale because of unit weights and hyperspherical features. To mitigate such a scaling issue, prior work for centralized training adopts either a pre-defined [14,74] or a learnable [23,31] scaling parameter τ in $\tau \cdot \mathbf{W}^k\mathbf{z}$ (*i.e.*, temperature) to stabilize optimization. Unfortunately, performing a grid search on the scaling hyper-parameter would add significant communication and computation overheads in the setting of federated learning. A freely learnable scaling parameter could also aggravate local models' disparity when the clients have different scaling parameters.

Interestingly, historical [19,63] and recent [4,27,34] studies show competitive results of mean square error (MSE) [10] compared with CE for classification tasks on modern deep architectures. We refer readers to [53] and the related literature [8,50,72] for a more in-depth theoretical discussion. In this work, we use MSE to learn hyperspherical features to bypass the scaling issue of CE, *i.e.*,

$$\mathcal{L}_{\text{MSE}}(\mathbf{W}^k\mathbf{z},\ y) = \frac{1}{C}\|\mathbf{W}^k\mathbf{z} - \texttt{one\_hot}(y)\|^2 = \frac{1}{C}\sum_{i=1}^{C}\left(\mathbf{w}_i^k\mathbf{z} - \mathbf{1}(i=y)\right)^2, \tag{5}$$

where $\mathbf{1}(i=y)$ is equal to 1 if and only if $i=y$, C is the number of classes, and $\texttt{one\_hot}(\cdot)$ is the one-hot vector representation of a label.

4.3 Fast Federated Calibration (FFC)

Throughout federated training of the hyperspherical representation, the weight of classifier $\mathbf{W}^k$ is fixed to help align learning targets of local features. So naturally, after federated training, a calibration on the classifier could be useful in improving the accuracy of the resulting global model. We, in turn, fix the learnt global feature extractor and calibrate the global classifier to boost the performance of the global model.

An interesting effect of using MSE loss in conjunction with a linear classifier is that we are able to calculate the unique closed-form optimum of classifier's weight matrix given the input of the linear classifier. Formally, the objective of calibrating the classifier $\mathbf{W}$ is,

$$\arg\min_{\mathbf{W}}\ \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathcal{L}_{\text{MSE}}\left(\mathbf{W}\mathbf{z},\ y\right)\right], \quad \text{where } \mathbf{z} = f_\theta(\mathbf{x})/\|f_\theta(\mathbf{x})\| \text{ and } \mathcal{D} = \bigcup\nolimits_{k\in[K]}\mathcal{D}^k \ . \tag{6}$$

We temporarily remove the superscripts of $(\theta, \mathbf{W})$ to emphasize that we consider the global feature extractor and global classifier. We refer to $\mathcal{D}$ as the whole dataset which consists of all local training data. Equation (6) is essentially a least square problem, which has a closed-form solution, *i.e.*,

$$\mathbf{W}^* = (\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top\mathbf{Y} \ . \tag{7}$$

求解得到的W

The i-th row of $\mathbf{Z}$ is the normalized feature vector $\mathbf{z}_i$ corresponding to the i-th sample $\mathbf{x}_i$ in $\mathcal{D}$. Similarly, the i-th row of $\mathbf{Y}$ is the one-hot target vector $\texttt{one\_hot}(y_i)$ corresponding to $\mathbf{x}_i$.

Obtaining $\mathbf{Z}$ (and $\mathbf{Y}$) requires all clients to upload their features (and corresponding labels). However, in the context of federated learning, sharing features and labels will cause prohibitively expensive communication overheads

and potential model inversion attack [17,93]. To achieve efficient and privacy-enhanced calibration, we propose to compute $\mathbf{W}^*$ in a distributed manner. It is inspired by that matrix multiplication can be implemented by a sum of outer products [46,92]. Take $\mathbf{Z}^\top\mathbf{Z}$ as an example. We can rewrite the $\mathbf{Z}^\top\mathbf{Z}$ as a sum of outer products between columns of $\mathbf{Z}^\top$ and rows of $\mathbf{Z}$ such that

$$\mathbf{Z}^\top\mathbf{Z} = \underbrace{n\,\mathbf{z}_1^\top \otimes n\,\mathbf{z}_1 + \cdots}_{\text{Client 1}} \cdots \underbrace{\cdots + \cdots}_{\text{Client k}} \cdots \underbrace{\cdots + n\,\mathbf{z}_{|\mathcal{D}|}^\top \otimes n\,\mathbf{z}_{|\mathcal{D}|}}_{\text{Client K}}. \tag{8}$$

FL场景下的W计算

As a result, to calculate $\mathbf{Z}^\top\mathbf{Z}$, each client can complete a fraction of sum over their local features (as shown in Eq. (8)) and upload the intermediate results to the sever to finish the computation, rather than upload private local features $\mathbf{z}_i$. In addition, the computation of $\mathbf{Z}^\top\mathbf{Y}$ in Eq. (7) can be calculated in the same way. See Fig. 1 (*Right*) for a vivid illustration.

Algorithm. We elaborate the FFC algorithm below:

1. On clients: Each client receives the latest global feature extractor f_θ from the server and computes the $\mathbf{V}^k \in \mathbb{R}^{l\times l}$, $\mathbf{U}^k \in \mathbb{R}^{l\times C}$ on its local data $\mathcal{D}^k$,
$$\mathbf{V}^k = \textstyle\sum_{i=1}^{|\mathcal{D}^k|} \mathbf{z}_i^\top\mathbf{z}_i, \quad \mathbf{U}^k = \sum_{i=1}^{|\mathcal{D}^k|} \mathbf{z}_i^\top \texttt{one\_hot}(y_i)\ , \tag{9}$$
where $\mathbf{z}_i = f_\theta(\mathbf{x}_i)/\|f_\theta(\mathbf{x}_i)\|$ and $(\mathbf{x}_i, y_i) \sim \mathcal{D}^k$.
2. On server: The server receives all $\{(\mathbf{V}^k, \mathbf{U}^k) \mid k \in [K]\}$ from clients and computes the closed-form weights optimum,
$$\mathbf{W}^* = \left(\textstyle\sum_{k=1}^{K} \mathbf{V}^k\right)^{-1} \left(\sum_{k=1}^{K} \mathbf{U}^k\right). \tag{10}$$

shape= 1 * C, W类似新添加的映射层

Table 1. Comparison of communication and computation costs for the classifier over 10 clients for 100 rounds. We assume that communicated weights are in 32-bit. The number of FLOPs is computed by considering both clients ("(C)") and the server ("(S)"). Since all approaches require forward of the classifier, we exclude it from calculation. "Every" (and "Once") means communicating at every round (and only once).

Method	Communication			Computation		Accuracy
	Objects	Frequency	Size (MB)	Operation	FLOPs (G)	
FedAvg	$\mathbf{W} \in \mathbb{R}^{l\times C}$	Every	102	Update $\mathbf{W}$ (S) Avg (S)	1.9×10^4	68.78
CCVR	$\mathbf{W} \in \mathbb{R}^{l\times C}$	Every	758	GMM fitting (C) Sampling&Tukey (S) Fine-tuning (S)	2.1×10^4	69.14
	$\mu \in \mathbb{R}^{l\times C}$ $\mathbf{\Sigma} \in \mathbb{R}^{l\times l\times C}$	Once				
SphereFed (Ours)	$\mathbf{U} \in \mathbb{R}^{l\times C}$ —$\mathbf{V} \in \mathbb{R}^{l\times l}$	Once	7	Eq. (9) (C) Eq. (10) (S)	1.7×10^2	71.85

Communication and Computation. FFC introduces much lighter communication and computation overhead to clients/server compared with state-of-the-art calibration methods like CCVR [48]. For instance, computing Eq. (9) on one

client requires $2l|\mathcal{D}^k|(l+C)$ FOLPs which is less than the total computation of the classifier's local training for one round. In addition, the communication amount of FFC is $l(l+C)$ parameters. In Table 1, we compare the communication and computation amount for the classifier of FFC against baselines (*e.g.*, FedAvg [51] and CCVR [48]) over 100 rounds with MobileNetV2 and CIFAR-100. As depicted in Table 1, SphereFed and FFC are more efficient than both FedAvg and CCVR in terms of communication and computation. In Sect. 5.4, we also provide latency comparison measured on a real embedded hardware.

5 Experiments

5.1 Experimental Setup

Baselines. Our proposed methods (*i.e.*, SphereFed and FFC) are compatible with and complementary to several existing federated learning algorithms like FedAvg [51], FedProx [43], FedNova [76], FedOpt [60] and so on. We refer to those algorithms as "base algorithms" in the remaining of this paper. We test five widely used models including a seven-layer ConvNet [85,90] and other modern deep architectures like MobileNetV2 [62], ResNet18 [22], VGG13 [36,67], SENet [26]. For all models, we refer to the last fully-connected layer as the classifier and all the other layers as the feature extractor.

Benchmarks. Following prior literature [40,41,84], we consider two representative and challenging image classification tasks for federated learning, CIFAR-100 [35] and TinyImageNet [37]. CIFAR-100 has 50,000 training samples from 100 classes. Prior method [48] obtains relatively small improvement against base algorithms on CIFAR-100 because the virtual features sampled from the class-wise Gaussian Mixture Model (GMM) are less separable when the number of class increases, thereby constraining its practicality for realistic applications. In this work, we show that our proposed methods are able to achieve superior performance for such many-class classification tasks. For example, we evaluate our methods on TinyImageNet which is larger than CIFAR-100 in terms of the input size, the number of samples, and the number of classes, as a more challenging dataset. Empirical results indicate that our proposed methods provide consistent improvements across multiple base algorithms, model architectures and datasets.

Like previous studies [20,30,41], we partition the training set of CIFAR-100 and TinyImageNet to K clients according to a Dirichlet distribution with a concentration parameter α to simulate the data distribution of federated learning. The default number of client is set to $K = 10$. A smaller concentration parameter will result in higher non-*i.i.d.* degree of partitioning. For example, when $\alpha = 0.1$, one client could have less than ten data samples in some classes. We consider three different non-*i.i.d.* degrees for CIFAR-100 to study how data heterogeneity degree impacts methods' performance. For fair comparison, we use exactly the same partitioning for all methods. The original test sets of CIFAR-100 and TinyImageNet are used to measure the resulted global model's testing accuracy.

Implementation. We use the SGD optimizer with a momentum 0.9 and a weight decay 10^{-5} for all approaches. Since we change the loss function from cross entropy to mean square error and these two loss functions have different magnitude, we tune the learning rate for both baselines and our methods using grid search. We note that SphereFed and FFC do not introduce any extra hyper-parameter to base algorithms. In addition, we observe that our methods are more robust to various learning rates than base algorithms in Sect. 5.3. For baselines with extra hyper-parameters, we either use the recommended values from their papers [48,76] or carefully tune them [43,60]. To tune hyper-parameters, we use a 15% of training data for validation. We train all approaches for 100 rounds and decay the learning rate every round using a cosine annealing schedule [47]. We use $B = 64$ local batch size and $E = 10$ local epochs unless otherwise stated. In Appendix A, we further test our methods on different federated learning settings by varying local training epochs, number of clients, clients' participating rate, and learning rate scheduling strategies. We observe similar trends as shown in the following sections.

Table 2. Accuracy (%) on CIFAR-100 and TinyImageNet with different degrees of non-*i.i.d.* . "+" means applying a considered method (CCVR, BABU and Ours) to a base FL algorithm (FedAvg, FedProx, FedNova and FedOpt). "↑" (and "↓") means accuracy improvement (and degradation) compared with the corresponding base algorithm.

Model	Method	IID	$\boldsymbol{\alpha = 0.5}$	$\boldsymbol{\alpha = 0.1}$	TinyImageNet
MobileNetV2	FedAvg	71.86	68.78	63.90	29.95
	+ CCVR	72.09 (↑ 0.23)	69.14 (↑ 0.36)	64.05 (↑ 0.15)	31.41 (↑ 1.46)
	+ BABU	71.84 (↓ 0.02)	69.35 (↑ 0.57)	64.91 (↑ 1.01)	28.38 (↓ 1.57)
	+ Ours	**73.56 (↑ 1.72)**	**71.85 (↑ 3.07)**	**66.52 (↑ 2.62)**	**34.72 (↑ 4.76)**
ResNet	FedProx	70.19	67.50	65.63	30.55
	+ CCVR	71.31 (↑ 0.12)	67.89 (↑ 0.39)	66.09 (↑ 0.46)	32.56 (↑ 2.01)
	+ BABU	71.66 (↑ 1.47)	69.62 (↑ 2.12)	67.90 (↑ 2.27)	31.87 (↑ 0.32)
	+ Ours	**73.41 (↑ 3.22)**	**72.20 (↑ 4.70)**	**69.19 (↑ 3.56)**	**35.21 (↑ 4.66)**
VGG13	FedNova	62.12	60.49	57.20	39.63
	+ CCVR	62.53 (↑ 0.41)	61.61 (↑ 1.12)	58.13 (↑ 0.93)	40.12 (↑ 0.49)
	+ BABU	62.03 (↓ 0.09)	60.54 (↑ 0.05)	58.95 (↑ 1.75)	40.87 (↑ 1.24)
	+ Ours	**65.50 (↑ 3.38)**	**65.12 (↑ 4.63)**	**62.54 (↑ 5.34)**	**45.21 (↑ 5.58)**
SENet	FedOpt	61.89	59.60	57.46	24.29
	+ CCVR	61.97 (↑ 0.08)	60.42 (↑ 0.82)	57.93 (↑ 0.47)	25.01 (↑ 0.72)
	+ BABU	62.27 (↑ 0.38)	59.69 (↑ 0.09)	56.75 (↓ 0.71)	25.34 (↑ 1.05)
	+ Ours	**65.15 (↑ 3.26)**	**65.69 (↑ 6.09)**	**62.61 (↑ 5.15)**	**29.84 (↑ 5.55)**

5.2 Results

We present in Table 2 the test accuracy of various base algorithms before and after applying our methods (*i.e.*, SphereFed and FFC) and two state-of-the-art baselines (*i.e.*, CCVR [48] and BABU [54]). Our proposed methods improve these base algorithms consistently across model architectures and datasets.

CCVR estimates Gaussian Mixture Model (GMM) for features on the class granularity on clients and samples virtual features from the GMM to fine-tune the classifier on the server. However, when the number of classes is relatively large (*e.g.*, 100 for CIFAR-100 and 200 for TinyImageNet), class-wise GMMs are not sufficiently separable to facilitate the fine-tuning. As a result, the improvement brought by CCVR is relatively small [48].

BABU [54] keeps the classifier fixed after random initialization during federated training and then fine-tunes on each client's local dataset individually for personalization. Two evaluation metrics are considered in BABU: (i) initial accuracy which is calculated with the single global model on the global test set and (ii) personalized accuracies measured with personalized models on local test sets over clients. As mentioned previously, we focus on the former (*i.e.*, initial accuracy) rather than personalized federated learning, while an analysis on how our method helps personalized federated learning is provided in Appendix B.

An interesting finding is that our methods tend to bring more accuracy gains when the non-*i.i.d.* degree is higher. This confirms our observation that a higher non-*i.i.d.* degree leads to more severe issues on classifier disparity and inconsistent local learning targets. With our methods, the performance gap between *i.i.d.* and non-*i.i.d.* data is reduced. For instance, the performance gap between "IID" and "$\alpha = 0.1$" is 4.92% for base algorithm FedNova, while this gap is reduced to 2.96% after applying the proposed approaches.

Table 3. Quantitative ablation study of Hyperspherical Federated Learning (SphereFed). We investigate the effectiveness of each design component by applying them individually using FedAvg as the base algorithm on MobileNetV2 and CIFAR-100 ($\alpha = 0.5$). "Fix (R)" ("Fix (OU)") means fixing the classifier with random (orthogonal and unit-norm) initialization. "Norm" represents normalizing features.

	FedAvg	+ Fix (R)	+ Fix (OU)	+ Norm	+ Fix (OU) +Norm
CE	68.78	69.35 (↑ 0.57)	69.65 (↑ 0.87)	69.76 (↑ 0.98)	70.87 (↑ 2.09)
MSE	66.42 (↓ 2.36)	67.05 (↓ 1.73)	67.21 (↓ 1.57)	Diverging	**71.85** (↑ **3.07**)

5.3 Ablation Studies

Besides overall effectiveness, we perform several ablation experiments which help understand the significance of each component in the proposed method.

The Importance of Ortho-normalization. In Table 3, we first compare different initializations of the fixed classifier. For the orthogonal and unit-norm

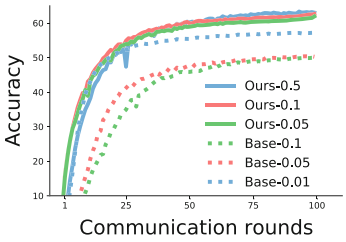

Fig. 4. The impact of different learning rates for "FedNova" and "FedNova + SphereFed". After applying SphereFed, training becomes more robust to different learning rates.

Table 4. Ablation study for FFC. Both CCVR and FFC methods exhibit performance gain on non-*i.i.d.* data. For sanity check, we collect local train sets to fine-tune the classifier. FFC is able to achieve a larger accuracy improvement than CCVR with significantly less communication and computation overheads (Table 1).

Calibration	IID	$\alpha = 0.1$
W/o	65.07	61.66
CCVR	65.03 (↓ 0.04)	62.09 (↑ 0.43)
FFC	**65.15 (↑ 0.08)**	**62.61 (↑ 0.95)**
Sanity check	65.17 (↑ 0.10)	62.64 (↑ 0.98)

initialization ("+ Fix (OU)"), we generate orthogonal weight matrix via the classic Gram-Schmidt process [1,58]. Other generation methods [52,71] are also considered in the appendix and no significant differences are observed. For the random initialization ("+ Fix (R)"), we instantiate it with He Initialization [21] which is the default initialization method in widely used packages such as PyTorch [56]. Random initialization achieves a comparable but slightly lower accuracy gain because two random vectors tend to be more orthogonal when their dimensionality increases [38,54], while orthogonal initialization directly ensures that.

In addition, SphereFed also normalizes features before feeding them to the classifier (denoted "Norm" in Table 3). After normalization, features are in the same unit hypersphere as the row vectors of classifier's weight and the feature extractor can focus on learning features' directions with the guidance of the fixed classifier. Applying feature normalization for the "Fix (OU)" completes the construction of hyperspherical representation and leads to about 1.22 accuracy gain. More importantly, we show that "Fix (OU)" and feature normalization work better with MSE than CE in the following discussion.

The Superiority of Mean-Square-Error Loss. We evaluate both CE and MSE loss functions in Table 3 to validate our choice of MSE loss. We confirm that replacing CE with MSE improves the accuracy by a considerable margin. The reasons are stated in Sect. 4.3 that MSE avoids the scaling issue and fully exploits the benefit of "Fix (OU) + Norm". For "CE + Fix (OU) + Norm", we find that it is quite sensitive to the scaling hyper-parameter (*i.e.*, temperature). Although we carefully tune the scaling factor τ and report the best result in Table 3, it is difficult and expensive to find the optimal τ in practice.

The Robustness of SphereFed Training. Compared with base algorithms, SphereFed does not introduce any extra hyper-parameters. Since the CE and MSE losses have different magnitudes, we tune their learning rates respectively from a set of candidate learning rates. Interestingly, we observe that SphereFed is more robust than the corresponding base algorithm. In Fig. 4, we test three differ-

ent learning rates for "FedNova" (as the base algorithm) and "FedNova + SphereFed" with VGG13 on CIFAR-100 ($\alpha = 0.1$). It is observed that "FedNova + SphereFed" is less sensitive to different learning rates.

How Beneficial is FFC? In this set of experiments, we investigate how the closed-form classifier calibration (*i.e.*, FFC) improves test accuracy. We apply CCVR and FFC individually for the classifier of SENet on CIFAR-100 after federated learning with "FedOpt + SphereFed". As a sanity check, we collect all local train sets to fine-tune the classifier only. To ensure the sanity check truly reveals the upper bound of classifier calibration, we experiment different loss functions (*i.e.*, CE and MSE) and learning rates for the sanity fine-tuning and report the best results we get. It can be seen from Table 4 that both CCVR and FFC achieve performance gains on non-*i.i.d.* data while FFC is able to improve accuracy more. CCVR estimates a GMM distribution for each class's features and sample virtual features for model fine-tuning on the server. However, such a class-wise method has relatively high communication and computation complexities (which scale linearly with number of classes). Moreover, GMMs of different classes could be less separable when the number of classes increases, thereby further limiting CCVR's effectiveness. In contrast, FFC, with provable formulations, is agnostic to number of classes and more suitable for realistic many-class federated learning tasks [25,81]. It is expected that FFC obtains comparable results as the sanity check because solving a linear classifier with either closed-form equations or SGD will converge to similar optimums [9,61].

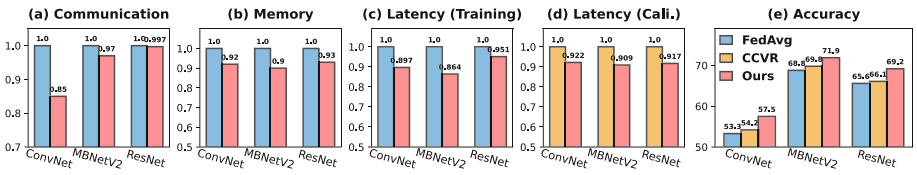

Fig. 5. Efficiency comparison measured on a neural network accelerator [78,91] with three models and CIFAR-100. (a) The normalized total communication amount. (b) The normalized peak memory consumption during local training. (c) The normalized latency for one-round local training. (d) The normalized latency for classifier calibration. (e) The test accuracy.

5.4 Efficient Communication and Computation

Besides accuracy gain, Hyperspherical Federated Learning also brings communication and computation savings depending on the size of classifier.

In Table 1, we compare the communication and computation costs related to the classifier for FedAvg, CCVR and our methods on MobileNetV2 and CIFAR-100. As seen in Table 1, SphereFed eliminates the need of updating and communicating (neither uploading nor downloading), resulting over two orders of magnitude communication and computation savings compared to FedAvg and CCVR.

Figure 5 depicts the relative computational savings of different approaches during the federated training process, measured on a DNN training accelerator built on Xilinx VC707 FPGA evaluation board [78,91]. Detailed settings about the DNN training accelerator are described in the appendix. SphereFed enables us to skip gradient computing for the classifier and thus to release some intermediate tensors used by gradient computing earlier. Overall, SphereFed achieves up to 10.5% and 13.6% savings on memory consumption and processing latency compared with FedAvg, respectively. SphereFed achieves a greater saving of training latency on MobileNetV2 than ConvNet and ResNet. This is because the computation workload associated with the convolutional layers are smaller in MobileNetV2 due to the usage of the depthwise separable operations [62], leading to a greater relative savings when the classifier is skipped during the local training. We also measure the latency of classifier calibration using CCVR and our Fast Federated Calibration in Fig. 5 (d). Since calibration-related computation happens on both clients and the server, the reported latency consists of both the average latency on one client and the on-server latency. Our closed-form calibration saves up to 9.1% latency against CCVR and this efficiency improvement will be more pronounced when number of classes increases as analysed above.

6 Conclusions

We presented the Hyperspherical Federated Learning (SphereFed) framework to address the non-*i.i.d.* issue. The proposed method focuses on the pre-learning phase and is complementary to existing federated learning methods. The hyperspherical representation is learned against a frozen classifier composed of orthonormal basis vectors, and the classifier is calibrated after training. We show that a mean squared loss is more suitable to hyperspherical representation as opposed to a cross-entropy loss due to the scaling issues. A Fast Federated Calibration (FFC) approach is proposed based on the mean squared loss. Extensive experiments indicate that SphereFed improves multiple existing federated learning algorithms by considerable margins.

Acknowledgements. This research was supported in part by the Air Force Research Laboratory under award number FA8750-18-1-0112.

References

1. Torch.linalg.qr. https://pytorch.org/docs/stable/generated/torch.linalg.qr.html
2. Acar, D.A.E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., Saligrama, V.: Federated learning based on dynamic regularization. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=B7v4QMR6Z9w
3. Acar, D.A.E., et al.: Debiasing model updates for improving personalized federated training. In: International Conference on Machine Learning, pp. 21–31. PMLR (2021)

4. Achille, A., Golatkar, A., Ravichandran, A., Polito, M., Soatto, S.: LQF: linear quadratic fine-tuning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15729–15739 (2021)
5. Al-Shedivat, M., Gillenwater, J., Xing, E., Rostamizadeh, A.: Federated learning via posterior averaging: a new perspective and practical algorithms. In: International Conference on Learning Representations (ICLR) (2021)
6. Anzai, Y.: Pattern Recognition and Machine Learning. Elsevier, Amsterdam (2012)
7. Arivazhagan, M.G., Aggarwal, V., Singh, A.K., Choudhary, S.: Federated learning with personalization layers. arXiv preprint arXiv:1912.00818 (2019)
8. Belkin, M.: Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. arXiv preprint arXiv:2105.14368 (2021)
9. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
10. Brier, G.W., et al.: Verification of forecasts expressed in terms of probability. Mon. Weather Rev. **78**(1), 1–3 (1950)
11. Bui, D., et al.: Federated user representation learning. arXiv preprint arXiv:1909.12535 (2019)
12. Chen, C., et al.: Communication-efficient federated learning with adaptive parameter freezing. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 1–11. IEEE (2021)
13. Chen, Y., Sun, X., Jin, Y.: Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. IEEE Trans. Neural Netw. Learn. Syst. **31**(10), 4229–4238 (2019)
14. Cheraghian, A., Rahman, S., Fang, P., Roy, S.K., Petersson, L., Harandi, M.: Semantic-aware knowledge distillation for few-shot class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2534–2543 (2021)
15. Collins, L., Hassani, H., Mokhtari, A., Shakkottai, S.: Exploiting shared representations for personalized federated learning. In: International Conference on Machine Learning, pp. 2089–2099. PMLR (2021)
16. Diao, E., Ding, J., Tarokh, V.: HeteroFL: computation and communication efficient federated learning for heterogeneous clients. In: International Conference on Learning Representations (2021)
17. Dong, X., Yin, H., Alvarez, J.M., Kautz, J., Molchanov, P.: Deep neural networks are surprisingly reversible: a baseline for zero-shot inversion. arXiv preprint arXiv:2107.06304 (2021)
18. Duan, J.-H., Li, W., Lu, S.: FedDNA: federated learning with decoupled normalization-layer aggregation for non-IID data. In: Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., Lozano, J.A. (eds.) ECML PKDD 2021. LNCS (LNAI), vol. 12975, pp. 722–737. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86486-6_44
19. Golik, P., Doetsch, P., Ney, H.: Cross-entropy vs. squared error training: a theoretical and experimental comparison. In: InterSpeech, vol. 13, pp. 1756–1760 (2013)
20. He, C., et al.: FedML: a research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518 (2020)
21. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

23. Hoffer, E., Hubara, I., Soudry, D.: Fix your classifier: the marginal value of training the last weight layer. In: International Conference on Learning Representations (2018)
24. Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 (2019)
25. Hsu, T.-M.H., Qi, H., Brown, M.: Federated visual classification with real-world data distribution. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12355, pp. 76–92. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58607-2_5
26. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
27. Hui, L., Belkin, M.: Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In: ICLR (2020)
28. Jain, V., Learned-Miller, E.: Online domain adaptation of a pre-trained cascade of classifiers. In: CVPR 2011, pp. 577–584. IEEE (2011)
29. Jiang, Y., Konečnỳ, J., Rush, K., Kannan, S.: Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488 (2019)
30. Kairouz, P., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
31. Kang, B., et al.: Decoupling representation and classifier for long-tailed recognition. In: International Conference on Learning Representations (2020)
32. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143. PMLR (2020)
33. Khosla, P., et al.: Supervised contrastive learning. Adv. Neural. Inf. Process. Syst. **33**, 18661–18673 (2020)
34. Kornblith, S., Chen, T., Lee, H., Norouzi, M.: Why do better loss functions lead to less transferable features? Adv. Neural Inf. Process. Syst. **34** (2021)
35. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report (2009)
36. Kuangliu: Pytorch-cifar/vgg.py. https://github.com/kuangliu/pytorch-cifar/blob/master/models/vgg.py
37. Le, Y., Yang, X.S.: Tiny imagenet visual recognition challenge (2015)
38. Lezama, J., Qiu, Q., Musé, P., Sapiro, G.: Ole: orthogonal low-rank embedding - a plug and play geometric loss for deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8109–8118 (2018)
39. Li, D., Wang, J.: FedMD: heterogenous federated learning via model distillation. In: NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality (2019)
40. Li, Q., Diao, Y., Chen, Q., He, B.: Federated learning on non-IID data silos: an experimental study. In: IEEE International Conference on Data Engineering (2021)
41. Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10713–10722 (2021)
42. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Sig. Process. Mag. **37**(3), 50–60 (2020)
43. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)

44. Liang, P.P., et al.: Think locally, act globally: federated learning with local and global representations. arXiv preprint arXiv:2001.01523 (2020)
45. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: SphereFace: deep hypersphere embedding for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 212–220 (2017)
46. Liu, X., Tang, Z., Huang, H., Zhang, T., Yang, B.: Multiple learning for regression in big data. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 587–594. IEEE (2019)
47. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
48. Luo, M., Chen, F., Hu, D., Zhang, Y., Liang, J., Feng, J.: No fear of heterogeneity: classifier calibration for federated learning with non-IID data. IN: 35th Conference on Neural Information Processing Systems (2021)
49. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(86), 2579–2605 (2008). http://jmlr.org/papers/v9/vandermaaten08a.html
50. Mai, X., Liao, Z.: High dimensional classification via empirical risk minimization: improvements and optimality. arXiv preprint arXiv:1905.13742 (2019)
51. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
52. Mettes, P., van der Pol, E., Snoek, C.: Hyperspherical prototype networks. Adv. Neural Inf. Process. Syst. **32** (2019)
53. Muthukumar, V., Narang, A., Subramanian, V., Belkin, M., Hsu, D., Sahai, A.: Classification vs regression in overparameterized regimes: does the loss function matter? J. Mach. Learn. Res. **22**(222), 1–69 (2021)
54. Oh, J., Kim, S., Yun, S.Y.: FedBABU: towards enhanced representation for federated image classification. In: International Conference on Learning Representations (2021)
55. Oh, J., Yoo, H., Kim, C., Yun, S.Y.: Boil: towards representation change for few-shot learning. In: International Conference on Learning Representations (2021)
56. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32** (2019)
57. Puigcerver, J., et al.: Scalable transfer learning with expert models. In: International Conference on Learning Representations (2021)
58. Pursell, L., Trimble, S.: Gram-Schmidt orthogonalization by gauss elimination. Am. Math. Mon. **98**(6), 544–549 (1991)
59. Raghu, A., Raghu, M., Bengio, S., Vinyals, O.: Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In: International Conference on Learning Representations (2019)
60. Reddi, S., et al.: Adaptive federated optimization. In: International Conference on Learning Representations (2021)
61. Saad, D.: Online algorithms and stochastic approximations. Online Learning **5**, 6–3 (1998)
62. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetv 2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
63. Sangari, A., Sethares, W.: Convergence analysis of two loss functions in soft-max regression. IEEE Trans. Sig. Process. **64**(5), 1280–1288 (2015)
64. Shamir, O., Srebro, N., Zhang, T.: Communication-efficient distributed optimization using an approximate newton-type method. In: International Conference on Machine Learning, pp. 1000–1008. PMLR (2014)

65. Shao, S., Xing, L., Wang, Y., Xu, R., Zhao, C., Wang, Y., Liu, B.: MHFC: multi-head feature collaboration for few-shot learning. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 4193–4201 (2021)
66. Shoham, N., et al.: Overcoming forgetting in federated learning on non-IID data. In: NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality (2019)
67. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
68. Singhal, K., Sidahmed, H., Garrett, Z., Wu, S., Rush, J., Prakash, S.: Federated reconstruction: partially local federated learning. Adv. Neural Inf. Process. Syst. **34** (2021)
69. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. Adv. Neural Inf. Process. Syst. **30** (2017)
70. Sun, B., Huo, H., Yang, Y., Bai, B.: PartialFED: cross-domain personalized federated learning via partial initialization. Adv. Neural Inf. Process. Syst. **34** (2021)
71. Tammes, P.M.L.: On the origin of number and arrangement of the places of exit on the surface of pollen-grains. Recueil des travaux botaniques néerlandais **27**(1), 1–84 (1930)
72. Thrampoulidis, C., Oymak, S., Soltanolkotabi, M.: Theoretical insights into multiclass classification: a high-dimensional asymptotic view. Adv. Neural Inf. Process. Syst. (2020)
73. Trefethen, L.N., Bau III, D.: Numerical Linear Algebra, vol. 50. SIAM (1997)
74. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
75. Venkat, N., Kundu, J.N., Singh, D., Revanur, A., et al.: Your classifier can secretly suffice multi-source domain adaptation. Adv. Neural. Inf. Process. Syst. **33**, 4647–4659 (2020)
76. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. Adv. Neural Inf. Process. Syst. (2020)
77. Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., Ramage, D.: Federated evaluation of on-device personalization. arXiv preprint arXiv:1910.10252 (2019)
78. Xilinx: Xilinx virtex-7 fpga vc707 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html
79. Xu, A., Huang, H.: Coordinating momenta for cross-silo federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 8735–8743 (2022)
80. Xu, C., Hong, Z., Huang, M., Jiang, T.: Acceleration of federated learning with alleviated forgetting in local training. In: International Conference on Learning Representations (2021)
81. Yang, K., Fan, T., Chen, T., Shi, Y., Yang, Q.: A quasi-newton method based vertical federated learning framework for logistic regression. In: The 2nd International Workshop on Federated Learning for Data Privacy and Confidentiality, in Conjunction with NeurIPS 2019 (2019)
82. Yao, X., Sun, L.: Continual local training for better initialization of federated models. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 1736–1740. IEEE (2020)
83. Ye, H.J., Hu, H., Zhan, D.C.: Learning adaptive classifiers synthesis for generalized few-shot learning. Int. J. Comput. Vision **129**(6), 1930–1953 (2021)

84. Yoon, J., Jeong, W., Lee, G., Yang, E., Hwang, S.J.: Federated continual learning with weighted inter-client transfer. In: International Conference on Machine Learning, pp. 12073–12086. PMLR (2021)
85. Yoon, T., Shin, S., Hwang, S.J., Yang, E.: Fedmix: approximation of mixup under mean augmented federated learning. In: International Conference on Learning Representations (2021)
86. You, K., Liu, Y., Wang, J., Long, M.: Logme: practical assessment of pre-trained models for transfer learning. In: International Conference on Machine Learning, pp. 12133–12143. PMLR (2021)
87. Yuan, H., Zaheer, M., Reddi, S.: Federated composite optimization. In: International Conference on Machine Learning, pp. 12253–12266. PMLR (2021)
88. Zhai, A., Wu, H.Y.: Classification is a strong baseline for deep metric learning. arXiv preprint arXiv:1811.12649 (2018)
89. Zhang, L., Luo, Y., Bai, Y., Du, B., Duan, L.Y.: Federated learning for non-IID data via unified feature learning and optimization objective alignment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4420–4428 (2021)
90. Zhang, S.Q., Lin, J., Zhang, Q.: A multi-agent reinforcement learning approach for efficient client selection in federated learning. arXiv preprint arXiv:2201.02932 (2022)
91. Zhang, S.Q., McDanel, B., Kung, H.: Fast: DNN training under variable precision block floating point with stochastic rounding. In: International Symposium on High-Performance Computer Architecture (2021)
92. Zhang, T., Yang, B.: Box-cox transformation in big data. Technometrics **59**(2), 189–201 (2017)
93. Zhao, N., Wu, Z., Lau, R.W., Lin, S.: What makes instance discrimination good for transfer learning? In: International Conference on Learning Representations (2021)
94. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. arXiv preprint arXiv:1806.00582 (2018)
95. Zheng, Y., Pal, D.K., Savvides, M.: Ring loss: convex feature normalization for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5089–5097 (2018)
96. Zhu, Y., Bai, Y., Wei, Y.: Spherical feature transform for deep metric learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12364, pp. 420–436. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58529-7_25
97. Zhu, Z., Hong, J., Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International Conference on Machine Learning, pp. 12878–12889. PMLR (2021)