FedBR: Improving Federated Learning on Heterogeneous Data via Local Learning Bias Reduction

Yongxin Guo [1 2] **Xiaoying Tang** [1 2 3] **Tao Lin** [4 5]

Abstract

Federated Learning (FL) is a way for machines to learn from data that is kept locally, in order to protect the privacy of clients. This is typically done using local SGD, which helps to improve communication efficiency. However, such a scheme is currently constrained by slow and unstable convergence due to the variety of data on different clients' devices. In this work, we identify three under-explored phenomena of biased local learning that may explain these challenges caused by local updates in supervised FL. As a remedy, we propose FedBR, a novel unified algorithm that reduces the local learning bias on features and classifiers to tackle these challenges. FedBR has two components. The first component helps to reduce bias in local classifiers by balancing the output of the models. The second component helps to learn local features that are similar to global features, but different from those learned from other data sources. We conducted several experiments to test FedBR and found that it consistently outperforms other SOTA FL methods. Both of its components also individually show performance gains. Our code is available at `https://github.com/lins-lab/fedbr`.

1. Introduction

Federated Learning (FL) is an emerging privacy-preserving distributed machine learning paradigm. The model is transmitted to the clients by the server, and when the clients have completed local training, the parameter updates are sent back to the server for integration. Clients are not required to provide local raw data during this procedure, maintaining their privacy. As the workhorse algorithm in FL, FedAvg (McMahan et al., 2016) proposes local SGD to improve communication efficiency. However, the considerable heterogeneity between local client datasets leads to inconsistent local updates and hinders convergence.

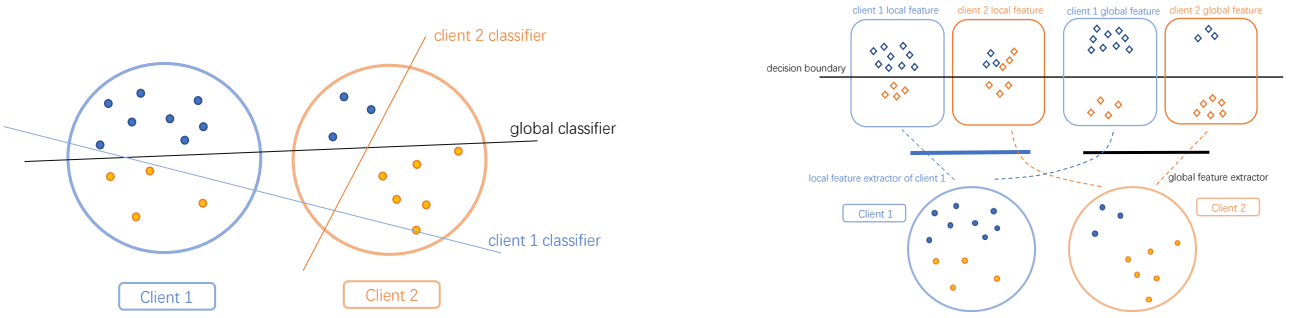
Several studies propose variance reduction methods (Karimireddy et al., 2019; Das et al., 2020), or suggest regularizing local updates towards global models (Li et al., 2018b; 2021) to tackle this issue. Almost all these existing works directly regularize models by utilizing the global model collected from previous rounds to reduce the variance or minimize the distance between global and local models (Li et al., 2018b; 2021). However, it is hard to balance the trade-offs between optimization and regularization to perform well, and data heterogeneity remains an open question in the community, as justified by the limited performance gain, e.g. in our Table 1 and experiment results in some previous works (Tang et al., 2022; Li et al., 2021; Yoon et al., 2021a; Chen & Chao, 2021; Luo et al., 2021).

Apart from the existing solutions, we revisit and reinterpret the fundamental issues in federated deep learning, caused by data heterogeneity and local updates. As our first contribution, we identify three pitfalls of FL systematically and in a unified view, termed *local learning bias*, from the perspective of representation learning[1]: 1) Biased local classifiers are unable to effectively classify unseen data (c.f. Figure 1(a)), due to the shifted decision boundaries dominated by local class distributions; 2) Local features (extracted by a local model) differ significantly from global features (similarly extracted by a centralized global model), even for the same input data (c.f. Figure 1(b)); and 3) Local features, even for data from different classes, are too close to each other to be accurately distinguished (c.f. Figure 1(b)). To this end, we propose FedBR, a unified method that leverages (1) a globally shared, label-distribution-agnostic pseudo-data and (2) two key algorithmic components, to simultaneously address the three difficulties outlined above. The first component of FedBR alleviates the first difficulty by forcing the output distribution of the pseudo-data to

[1]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China. [2]The Shenzhen Institute of Artificial Intelligence and Robotics for Society [3]The Guangdong Provincial Key Laboratory of Future Networks of Intelligence, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, China [4]Research Center for Industries of the Future, Westlake University [5]School of Engineering, Westlake University. Correspondence to: Xiaoying Tang <tangxiaoying@cuhk.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Please refer to section 3 for more justifications about the existence of our observations.



(a) Bias in local classifier. (b) Bias in local feature.

Figure 1: **Observation for learning bias of FL** on heterogeneous data with local updates. There are two clients in the figure, and each has two classes of data (red and blue points). **Biased local classifier** in 1(a): Client 1's decision boundary cannot accurately classify data samples from Client 2. **Biased local feature** in 1(b): The difference between features extracted by Client 1's local and global feature extractor is sustainable large. However, Client 2's local feature is close enough to Client 1's, even for input data from different data distributions/clients.

be balanced. The second component of FedBR aims to address the second and third difficulties simultaneously, where we develop a ***min-max contrastive learning method*** to learn client invariant local features. More precisely, a key two-stage algorithm is proposed to maximize the elimination of feature learning biases: the first stage learns a projection space to distinguish the features of two types, while the second stage enforces learned features on the projected feature space that are farther from local features and closer to global ones. All these can be unified into a simple yet effective min-max procedure to alleviate the local learning bias in FL, with trivial requirements on pseudo-data while still preserving privacy.

Our main contributions are:

- We provide a unified view to interpret the learning difficulty in FL with heterogeneous data, and identify three key pitfalls to explain the issue of local learning biases.
- We propose FedBR, a unified algorithm that leverages pseudo-data to reduce the learning bias on local features and classifiers. We design two orthogonal key components of FedBR to complement each other to improve the learning quality of clients with heterogeneous data.
- FedBR considerably outperforms other FL baselines by a large margin, as justified by extensive numerical evaluation on RotatedMNIST, CIFAR10, and CIFAR100. Besides, FedBR does not require the labeled or large number of global shared pseudo-data, thereby improving the efficiency.

2. Related Works

Federated Learning (FL). As the de facto FL algorithm, McMahan et al. (2016); Lin et al. (2020b) propose to use local SGD steps to alleviate the communication bottleneck. However, the objective inconsistency caused by the local data heterogeneity considerably hinders the convergence of FL algorithms (Li et al., 2018b; Wang et al., 2020b; Karimireddy et al., 2019; 2020; Guo et al., 2021). To address the issue of heterogeneity in FL, a series of projects has been proposed. FedProx (Li et al., 2018b) incorporates a proximal term into local objective functions to reduce the gap between the local and global models. SCAFFOLD (Karimireddy et al., 2019) adopts the variance reduction method on local updates, and Mime (Karimireddy et al., 2020) increases convergence speed by adding global momentum to global updates. Recently, Moon (Li et al., 2021) has proposed to employ contrastive loss to reduce the distance between global and local features. However, their projection layer is only used as part of the feature extractor, and cannot contribute to distinguishing the local and global features—a crucial step identified by our investigation for better model performance. In this paper, we focus on improving the global model in Federated Learning (FL) by designing methods that perform well on all local distributions. The designed algorithm works on the local training stage, which aligns with previous research in this area, such as McMahan et al. (2016); Li et al. (2018b); Karimireddy et al. (2019); Li et al. (2021); Tang et al. (2022). Other topics like improving the global aggregation stages (Wang et al., 2020a; Yoshida et al., 2019) or Personalized Federated Learning (PFL) methods (Tan et al., 2022; Wu et al., 2022; Jiang & Lin, 2023) are orthogonal to our approach and could be further combined with our method.

Data Augmentation in FL. To reduce data heterogeneity, some data-based approaches suggest sharing a global dataset among clients and combining global datasets with local datasets (Tuor et al., 2021; Yoshida et al., 2019). Some knowledge distillation-based methods also require a global dataset (Lin et al., 2020a; Li & Wang, 2019), which is used to transfer knowledge from local models (teachers) to global models (students). Considering the impractical of sharing the global datasets in FL settings, some recent research use proxy datasets with augmentation techniques. Astraea (Duan et al., 2019) uses local augmentation to create a globally balanced distribution. XorMixFL (Shin et al.,

2020) encodes a couple of local data and decodes it on the server using the XOR operator. FedMix (Yoon et al., 2021b) creates the privacy-protected augmentation data by averaging local batches and then applying Mixup in local iterations. VHL (Tang et al., 2022) relies on the created virtual data with labels and forces the local features to be close to the features of same-class virtual data. Different from previous works, this paper designs methods that utilize label-agnostic pseudo-data, and outperform other methods using significantly less pseudo-data.

3. The Pitfalls of FL on Heterogeneous Data

FL and local SGD. FL is an emerging learning paradigm that supposes learning on various clients while clients can not exchange data to protect users' privacy. Learning occurs locally on the clients, while the server collects and aggregates gradient updates from the clients. The standard FL considers the following problem:

$$f^* = \min_{\boldsymbol{\omega} \in R^d} \left[f(\boldsymbol{\omega}) = \sum_{i=1}^{N} p_i f_i(\boldsymbol{\omega}) \right], \qquad (1)$$

where $f_i(\boldsymbol{\omega})$ is the local objective function of client i, and p_i is the weight for $f_i(\boldsymbol{\omega})$. In practice, we set $p_i = {}^{|D_i|}/_{|D|}$ by default, where D_i is the local dataset of client i and D is the combination of all local datasets. The global objective function $f(\boldsymbol{\omega})$ aims to find $\boldsymbol{\omega}$ that can perform well on all clients. In the training process of FL, the communication cost between client and server has become an essential factor affecting the training efficiency. Therefore, local SGD (McMahan et al., 2016) has been proposed to reduce the communication round. In local SGD, clients perform multiple local steps before synchronizing to the server in each communication round.

Bias caused by local updates. In this paper, we consider improving previous works by proposing a label-agnostic method, and we first identify the pitfalls of FL on heterogeneous data in a label-agnostic way as follows.

Proposition 3.1 (Local Learning Bias in FedAvg)**.** *For FedAvg, the local models after local epochs could be biased, in detail,*

- Biased local feature: *For local feature extractor $F_i(\cdot)$, and centralized trained global feature extractor $F_g(\cdot)$, we have: 1) Given the data input X, $F_i(X)$ could deviate largely from $F_g(X)$. 2) Given the input from different data distributions X_1 and X_2, $F_i(X_1)$ could be very similar or almost identical to $F_i(X_2)$.*
- Biased local classifier: *After a sufficient number of iterations, local models classify all samples into only the classes that appeared in the local datasets.*

To verify the correctness of Proposition 3.1, we can use some toy examples to show the existence of the biased local feature and classifiers. For toy examples on more complex scenarios and on the benefits of using FedBR please refer to Appendix C.3.

Example 3.2 (Observation for biased local features)**.** *Figures 2(a) and 2(b) show that* local features differ from global features for the same input*, and Figures 2(b) and 2(c) show that* local features are similar even for different input distributions. *We define this observation as the "biased local feature". In detail, we calculate $F_1(X_1)$, $F_1(X_2)$, $F_g(X_1)$, and $F_g(X_2)$, and use t-SNE to project all the features to the same 2D space. We can observe that the local features of data in X_2 are so close to local features of data in X_1, and it is non-trivial to tell which category the current input belongs to by merely looking at the local features.*

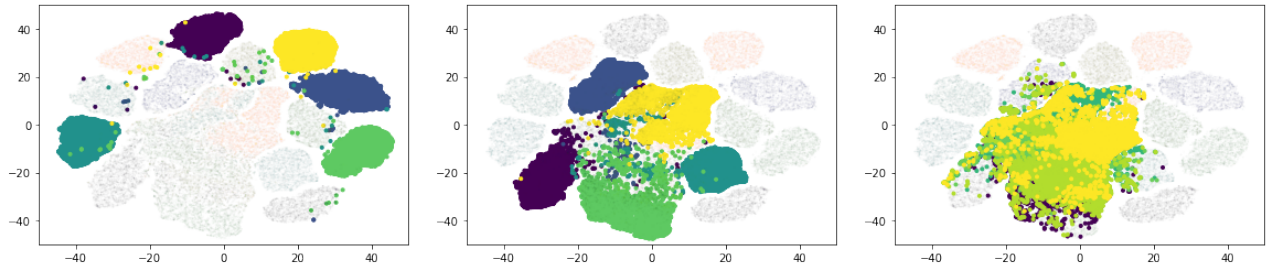
Example 3.3 (Observation for biased local classifiers)**.** *Figure 3 shows the output of the local model on data X_2, where all data in X_2 are incorrectly categorized into classes 0 to 4 of X_1. The observation, i.e. data from classes that are absent from local datasets cannot be correctly classified by the local classifiers, refers to the "biased local classifiers". More precisely, Figure 3(a) shows the prediction result of one sample (class 8) and Figure 3(b) shows the predicted distribution of all samples in X_2.*

Distinct from the local learning bias in previous works. We acknowledge the discussion of learning bias in some previous works, e.g. in Karimireddy et al. (2019); Li et al. (2018b; 2021). However, our work differs in several ways:

1. FedProx (Li et al., 2018b) defines local drifts as the differences in model weights, while SCAFFOLD (Karimireddy et al., 2019) considers gradient differences as client drifts. These methods, though have been effective on traditional optimization tasks, may only have marginal improvements on deep models, as shown in Tang et al. (2022); Li et al. (2021); Yoon et al. (2021a); Chen & Chao (2021); Luo et al. (2021).
2. MOON (Li et al., 2021) minimizes the distance between global and local features, but its performance is limited because they use only the projection layer as part of the feature extractor, and the contrastive loss diminished without our designed max step (c.f. Table 1 and Table 4).
3. VHL (Tang et al., 2022) defines local learning bias as the shift in features between samples of the same classes; however, this approach requires prior knowledge of local label information and results in a much larger virtual dataset, especially when increasing the number of classes. Our method instead achieves better performance with significantly fewer pseudo-data (see Table 2).

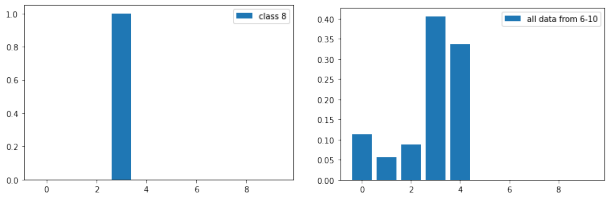
4. FedBR: Reducing Learning Bias in FL

Addressing the local learning bias is crucial to improving FL on heterogeneous data, due to the *bias* discussed in Proposition 3.1. To this end, we propose FedBR as shown in Figure 4, a novel framework that leverages the globally shared pseudo-data with two key components to reduce the local training bias, namely 1) reducing the local classifier's bias by balancing the output distribution of classifiers (com-



(a) Global feature of X_1, $F_g(X_1)$ (b) Local feature of X_1, $F_1(X_1)$ (c) Local feature of X_2, $F_1(X_2)$

Figure 2: **Observation for biased local features** on a shared t-SNE projection space. *Local updates will cause:* • *Large difference in local and global features for the same input data.* Colored points in sub-figures (a) & (b) denote the global and local features of data from X_1, and the same color indicates data from the same class. Notice that even for data from the same class (same color), the global and local features are clustered into two distinct groups, implying a considerable distance between global and local features even for the same input data distribution. • *High similarity of local features for different inputs.* Notice from sub-figure (b) & (c) that X_1 and X_2 are two disjoint datasets (no data from the same class). However, the local features of X_1 and X_2 are clustered into the same group by t-SNE, indicating the relatively small distance between local features of different classes. Results on mild conditions and different training stages can be found in Appendix C.



(a) Sample from class 8 (b) All samples in X_2

Figure 3: **Observation for biased local classifiers**: *the output distribution of the local classifiers will be dominated by the local class distribution.* The model is trained on data X_1 and tested on data X_2. The sub-figure (a) illustrates the model output distribution of a sample belonging to Class 8. The sub-figure (b) shows the total prediction distribution of all samples in X_2. Results show that *the biased local model will classify all samples into classes that are only present in the* X_1.

ponent 1), and 2) an adversary contrastive scheme to learn unbiased local features (component 2).

4.1. Overview of the FedBR

The learning procedure of FedBR on each client i involves the construction of a global pseudo-data (c.f. Section 4.2), followed by applying two key debias steps in a ***min-max*** approach to jointly form two components (c.f. Section 4.3 and 4.4) to reduce the bias in the classifier and feature, respectively.

The min-max procedure of FedBR can be interpreted as first projecting features onto spaces that can distinguish global and local feature best, then 1) minimizing the distance between the global and local features of pseudo-data and maximizing distance between local features of pseudo-data and local data; 2) minimize classification loss of both local data and pseudo-data:

Max Step: $\max_{\boldsymbol{\theta}} \mathcal{L}_{adv}(D_p, D_i)$

$$:= \mathbb{E}_{\mathbf{x}_p \sim D_p, \mathbf{x} \sim D_i} \left[\mathcal{L}_{con}(\mathbf{x}_p, \mathbf{x}, \boldsymbol{\phi}_g, \boldsymbol{\phi}_i, \boldsymbol{\theta}) \right] . \quad (2)$$

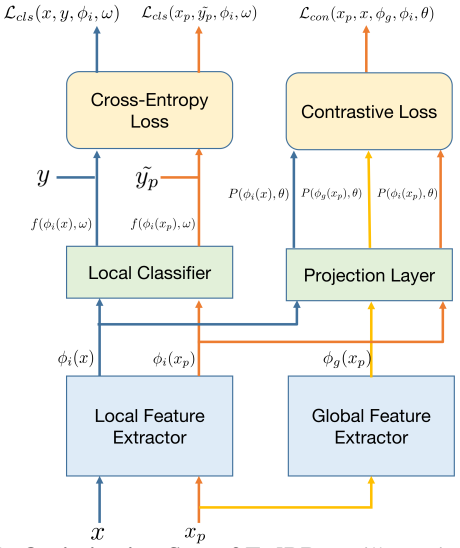


Figure 4: **Optimization flow of FedBR**: an illustration of how the three terms in (2) and (3) are calculated. We calculate the cross-entropy loss of local data $(\mathbf{x}, \mathbf{y})$, and pseudo-data $(\mathbf{x}_p, \tilde{\mathbf{y}_p})$, and use the local feature $\boldsymbol{\phi}_i(\mathbf{x}), \boldsymbol{\phi}_i(\mathbf{x}_p)$, and global feature $\boldsymbol{\phi}_g(\mathbf{x}_p)$ for contrastive loss.

Min Step: $\min_{\boldsymbol{\phi}_i, \boldsymbol{\omega}} \mathcal{L}_{gen}(D_p, D_i)$

$$\begin{aligned} := & \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim D_i} \left[\mathcal{L}_{cls}(\mathbf{x}, \mathbf{y}, \boldsymbol{\phi}_i, \boldsymbol{\omega}) \right] \\ & + \lambda \mathbb{E}_{\mathbf{x}_p \sim D_p} \left[\mathcal{L}_{cls}(\mathbf{x}_p, \tilde{\mathbf{y}_p}, \boldsymbol{\phi}_i, \boldsymbol{\omega}) \right] \\ & + \mu \mathbb{E}_{\mathbf{x}_p \sim D_p, \mathbf{x} \sim D_i} \left[\mathcal{L}_{con}(\mathbf{x}_p, \mathbf{x}, \boldsymbol{\phi}_g, \boldsymbol{\phi}_i, \boldsymbol{\theta}) \right] . \end{aligned} \quad (3)$$

$\mathcal{L}_{cls}$ and $\mathcal{L}_{con}$ represent the cross-entropy loss and a contrastive loss (will be detailed in Section 4.4), respectively. D_i denotes the distribution of the local dataset at client i. D_p is that of shared pseudo-dataset, where $\tilde{\mathbf{y}_p}$ is the pseudo-label of pseudo-data. The model is composed of a feature extractor $\boldsymbol{\phi}$ and a classifier $\boldsymbol{\omega}$, where the omitted subscript i and g correspond to the local client i and global parameters,

respectively (e.g. ϕ_g denotes the feature extractors received from the server at the beginning of each communication round). We additionally use a projection layer $\boldsymbol{\theta}$ for the max step to project features onto spaces where global and local features have the largest dissimilarity.

Apart from the cross-entropy loss of local data in (3), the second term aims to overcome the biased local classifier while the local feature is debiased by the third term.

The proposed FedBR is summarized in Algorithm 1. The global communication part is the same as FedAvg, and the choice of synchronizing the new pseudo-data to clients in each round is optional[2].

The benefit of FedBR for using less prior information. In additional to the superior performance, the design of FedBR has the following benefits: 1) Unlike previous works (Tang et al., 2022), our method does not require knowledge of the local label distribution and is label-agnostic. This means that the size of the pseudo-data will not increase as the number of classes increases. Our results, shown in Table 2 and Figure 6(b), demonstrate that our method, FedBR, can achieve better performance with significantly less pseudo-data. 2) Pseudo-data can be used for various tasks. Pseudo-data created using CIFAR10 performs well on tasks with local data from CIFAR10 and CIFAR100, as seen in Figure 6(c).

4.2. Construction of the Pseudo-Data

The choice of the pseudo-data in our FedBR framework is arbitrary. For ease of presentation and taking the communication cost into account, we showcase two construction approaches below and detail their performance gain over all other existing baselines in Section 5:

- **Random Sample Mean (RSM)**. Similar to the treatment in FedMix (Yoon et al., 2021b), one RSM sample of the pseudo-data is estimated through a weighted combination of a random subset of local samples, and the pseudo-label is set[3] to $\tilde{\mathbf{y}_p} = \frac{1}{C} \cdot \mathbf{1}$. It is worth noting that RSM *does not require the local data to be balanced* when constructing the pseudo-data, as long as the local data is distinct from the pseudo-data. We show in Figure 6(d) that our algorithm (FedBR) can achieve comparable performance using pseudo-data constructed from data with unbalanced label distribution. For more details, see Algorithm 2 in the appendix.
- **Mixture of local samples and the sample mean of a proxy dataset (Mixture)**. This strategy relies on applying the procedure of RSM to irrelevant and globally shared proxy data (refer to Algorithm 3). To guard the distribution distance between the pseudo-data and local data, one sample of the pseudo-data at each client is constructed by

$$\tilde{\mathbf{x}}_p = \frac{1}{K+1}(\mathbf{x}_p + \sum_{k=1}^{K} \mathbf{x}_k), \tilde{\mathbf{y}}_p = \frac{1}{K+1}(\frac{1}{C} \cdot \mathbf{1} + \sum_{k=1}^{K} \mathbf{y}_k), \quad (4)$$

where $\mathbf{x}_p$ is one RSM sample of the global proxy dataset, and $\mathbf{x}_k$ and $\mathbf{y}_k$ correspond to the data and label of one local sample (vary depending on the client). K is a constant that controls the closeness between the distribution of pseudo-data and local data. As we will show in Section 5, setting $K = 1$ is data-efficient yet sufficient to achieve good results.

[2]As shown in Figure 6(b), the communication-efficient variant of FedBR—i.e. only transferring pseudo-data at the beginning of the FL training—is on par with the choice of frequent pseudo-data synchronization.

[3]We assume that pseudo-data does not belong to any particular classes, and should not give high confidence to any of that.

Algorithm 1 Algorithm Framework of FedBR

Require: Local datasets $D_1, \ldots, D_N$, pseudo dataset D_p where $|D_p| = B$, and B is the batch size, number of local iterations K, number of communication rounds T, number of clients chosen in each round M, weights used in designed loss λ, μ, local learning rate η.
Ensure: Trained model $\boldsymbol{\omega}_T$, $\boldsymbol{\theta}_T$, $\boldsymbol{\phi}_T$.

```
Initialize ω_0, θ_0, φ_0.
for t = 0, ..., T − 1 do
    Send ω_t, θ_t, φ_t, D_p (optional) to all clients.
    for chosen client i = 1, ..., M do
        ω_i^0 = ω_t, θ_i^0 = θ_t, φ_i^0 = φ_t, φ_g = φ_t
        for k = 1, ..., K do
            # Max Step
            θ_i^k = θ_i^{k−1} + η∇_θ L_adv.
            # Min Step
            ω_i^k = ω_i^{k−1} − η∇_ω L_k.
            φ_i^k = φ_i^{k−1} − η∇_φ L_gen.
        Send ω_i^K, θ_i^K, φ_i^K to server.
    ω_{t+1} = (1/M) Σ_{i=1}^M ω_i^K.
    θ_{t+1} = (1/M) Σ_{i=1}^M θ_i^K.
    φ_{t+1} = (1/M) Σ_{i=1}^M φ_i^K.
```

Remark: preserving privacy via Mixture. The RSM method is similar to the data augmentation method used in FedMix. Similar to the discussion in FedMix, such a scheme may leak privacy. To address this, we propose using the Mixture as a privacy-preserving method. Mixture can even outperform RSM as justified in Figure 6(c).

4.3. Component 1: Reducing Bias in Local Classifiers

Due to the issue of label distribution skew or the absence of some samples for the majority/minority classes, the trained local model classifier tends to overfit the locally presented classes, and may further hinder the quality of the feature extractor (as justified in Figure 3 and Proposition 3.1).

As a remedy, here we implicitly mimic the global data distribution—by using the pseudo-data constructed in Section 4.2—to regularize the outputs and thus debias the classifier (note that Component 1 is the second term of (3)):

$$\lambda \mathbb{E}_{\mathbf{x}_p \sim D_i} \left[\mathcal{L}_{cls}(\mathbf{x}_p, \tilde{\mathbf{y}_p}, \boldsymbol{\phi}_i, \boldsymbol{\omega}) \right] .$$

4.4. Component 2: Reducing Bias in Local Features

In addition to alleviating the biased local classifier in Section 4.3, here we introduce a crucial adversary strategy to learn unbiased local features.

Intuition of constructing an adversarial problem. As discussed in Proposition 3.1, effective federated learning on heterogeneous data requires learning debiased local feature extractors that 1) can extract local features that are close to global features of the same input data; 2) can extract different local features for input samples from different distributions. However, existing methods that directly minimize the distance between global features and local features (Li et al., 2018b; 2021) have limited performance gain (c.f. Table 1) due to the diminishing optimization objective caused by the indistinguishability between the global and local features of the same input. To this end, we propose to extend the idea of adversarial training to our FL scenarios:

1. We construct a projection layer as the critical step to distinguish features extracted by the global and local feature extractor: such layer ensures that the projected features extracted by the local feature extractor will be close to each other (even for distinct local data distributions), but the difference between features extracted by the global and local feature extractor after projection will be considerable (even for the same input samples).
2. We can find that constructing such a projection layer can be achieved by maximizing the local feature bias discussed in Proposition 3.1. More precisely, it can be achieved by maximizing the distance between global and local features of pseudo-data and simultaneously minimizing the distance between local features of pseudo-data and local data.
3. We then minimize the local feature biases (discussed in Proposition 3.1) under the trained projection space, to enforce the learned local features of pseudo-data to be closer to the global features of pseudo-data but far away from the local features of real local data.

On the importance of utilizing the projection layer to construct the adversary problem. To construct the aforementioned adversarial training strategy, we consider using an additional projection layer to map features onto the projection space[4]. In contrast to the existing works that similarly add a projection layer (Li et al., 2021), we show that 1) simply adding a projection layer as part of the feature extractor has trivial performance gain (c.f. Figure 6(a)); 2) our design is the key step to reducing the feature bias and boosting the federated learning on heterogeneous data (c.f. Table 4).

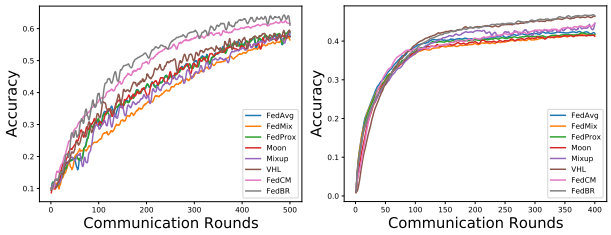
Objective function design. We extend the idea of Li et al. (2021) and improve the contrastive loss initially proposed in simCLR (Chen et al., 2020) to our challenging scenario. Different from previous works, we use the projected features (global and local) on pseudo-data as the positive pairs and rely on the projected local feature of both pseudo-data and local data as the negative pairs:

$$f_1 = \exp\left(\frac{\text{sim}\left(P_{\boldsymbol{\theta}}(\boldsymbol{\phi}_i(\mathbf{x}_p)), P_{\boldsymbol{\theta}}(\boldsymbol{\phi}_g(\mathbf{x}_p))\right)}{\tau_1}\right), \tag{5}$$

$$f_2 = \exp\left(\frac{\text{sim}\left(P_{\boldsymbol{\theta}}(\boldsymbol{\phi}_i(\mathbf{x}_p)), P_{\boldsymbol{\theta}}(\boldsymbol{\phi}_i(\mathbf{x}))\right)}{\tau_2}\right), \tag{6}$$

$$\mathcal{L}_{con}(\mathbf{x}_p, \mathbf{x}, \boldsymbol{\phi}_g, \boldsymbol{\phi}_i, \boldsymbol{\theta}) = -\log\left(\frac{f_1}{f_1 + f_2}\right), \tag{7}$$

where $P_{\boldsymbol{\theta}}$ is the projection layer parameterized by $\boldsymbol{\theta}$, τ_1 and τ_2 are temperature parameters, and sim is the cos-similarity function. Our implementation uses a tied value for τ_1 and τ_2 for the sake of simplicity, but an improved performance may be observed by tuning these two.



(a) CIFAR10 Convergence (b) CIFAR100 Convergence

Figure 5: **Convergence curve of algorithms on different datasets.** We split RotatedMNIST, CIFAR10, and CIFAR100 datasets to 10 clients, and report the mean accuracy on all local test datasets for each communications rounds. More Details refer to Figure 9 of Appendix C.

5. Experiments

5.1. Experiment Setting

We elaborate on experiment settings in Appendix A.

Baseline algorithms. We compare FedBR with both SOTA FL baselines including FedAvg (McMahan et al., 2016), Moon (Li et al., 2021), FedProx (Li et al., 2018b), VHL (Tang et al., 2022), FedMix (Yoon et al., 2021b), FedNTD (Lee et al., 2022), FedCM (Xu et al., 2021), and FedDecorr (Shi et al., 2023) which are most relevant to our proposed algorithms. Similar to a very recent study in benchmarking FL (Bai et al., 2023), we also contain domain generalization (DG) methods as baselines and check their performance under standard FL settings. For DG baselines, we choose GroupDRO (Sagawa et al., 2019), Mixup (Yan et al., 2020), and DANN (Ganin et al., 2015). We also discuss other DG baselines in Appendix A. Unless specially mentioned, all algorithms use FedAvg as the backbone algorithm.

Models and datasets. We examine all algorithms on RotatedMNIST, CIFAR10, and CIFAR100 datasets. We use a four-layer CNN for RotatedMNIST, VGG11 for CIFAR10, and Compact Convolutional Transformer (CCT (Hassani

[4]Such a projection layer is not part of the feature extractor or used for classification, as shown in Figure 4.

Table 1: **Performance of algorithms.** We split RotatedMNIST, CIFAR10, and CIFAR100 to 10 clients with $\alpha = 0.1$, and ran 1000 communication rounds on RotatedMNIST and CIFAR10 for each algorithm, 800 communication rounds CIFAR100. We report the mean of maximum (over rounds) 5 test accuracies and the number of communication rounds to reach the threshold accuracy.

Algorithm	RotatedMNIST (CNN)		CIFAR10 (VGG11)		CIFAR100 (CCT)	
	Acc (%)	Rounds for 80%	Acc (%)	Rounds for 55%	Acc (%)	Rounds for 43%
Local	14.67	-	10.00	-	1.31	-
FedAvg	82.47	828 (1.0X)	58.99	736 (1.0X)	44.00	550 (1.0X)
FedProx	82.32	824 (1.0X)	59.14	738 (1.0X)	43.09	756 (0.7X)
Moon	82.68	864 (0.9X)	58.23	820 (0.9X)	42.87	766 (0.7X)
DANN	84.83	743 (1.1X)	58.29	782 (0.9X)	41.83	-
GroupDRO	80.23	910 (0.9X)	56.57	835 (0.9X)	44.34	444 (1.2X)
FedBR (Ours)	**86.58**	**628 (1.3X)**	64.65	496 (1.5X)	45.14	352 (1.5X)
FedAvg + Mixup	82.56	840 (1.0X)	58.57	826 (0.9X)	46.37	358 (1.6X)
FedMix	81.33	902 (0.9X)	57.37	872 (0.8X)	42.69	-
FedBR + Mixup (Ours)	83.42	736 (1.1X)	**65.32**	**392 (1.9X)**	**47.75**	294 (1.9X)

Table 2: **Comparison with VHL.** We split CIFAR10 and CIFAR100 to 10 clients with $\alpha = 0.1$, and report the mean of maximum (over rounds) 5 test accuracies and the number of communication rounds to reach the threshold accuracy. We set different numbers of virtual data to check the performance of VHL, and pseudo-data only transfer once in FedBR (32 pseudo-data). For CIFAR100, we choose Mixup as the backbone.

Algorithm	CIFAR10 (VGG11)		CIFAR100 (CCT)	
	Acc (%)	Rounds for 60%	Acc (%)	Rounds for 46%
VHL (2000 virtual data)	61.23	886 (1.0X)	46.80	630 (1.0X)
VHL (20000 virtual data)	59.65	998 (0.9X)	46.51	714 (0.9X)
FedBR (32 pseudo-data)	**64.61**	**530 (1.8X)**	**47.67**	**554 (1.1X)**

Table 3: **Combining FedBR with other baselines.** We split CIFAR10 and CIFAR100 to 10 clients with $\alpha = 0.1$, and report the mean of maximum (over rounds) 5 test accuracies. For FedBR, pseudo-data only transfer once (32 pseudo-data) using **Mixture**. For CIFAR100, we choose Mixup as the backbone.

Algorithm	CIFAR10 (VGG11)		CIFAR100 (CCT)	
	w/o FedBR	+ FedBR	w/o FedBR	+ FedBR
FedAvg	58.99	64.66 (+5.67)	46.37	47.98 (+1.61)
FedCM	62.63	**65.32 (+2.69)**	46.15	46.95 (+0.80)
FedDecorr	54.15	62.70 (+8.55)	47.18	**48.34 (+1.16)**
FedNTD	59.10	59.26 (+0.16)	47.02	47.18 (+0.16)

et al., 2021)) for CIFAR100. We split the datasets following the idea introduced in (Yurochkin et al., 2019; Hsu et al., 2019; Reddi et al., 2021), where we leverage the Latent Dirichlet Allocation (LDA) to control the distribution drift with parameter α. The pseudo-data is chosen as ***RSM*** by default, and we also provide results on other types of pseudo-data (c.f. Figure 6(c)). By default, we generate one batch of pseudo-data (64 for MNIST and 32 for other datasets) in each round, and we also investigate only generating one batch of pseudo-data at the beginning of training to reduce the communication cost (c.f. Figure 6(b), Figure 6(c)). We use SGD optimizer and set the learning rate to 0.001 for RotatedMNIST, and 0.01 for other datasets. The local batch size is set to 64 for RotatedMNIST, and 32 for other datasets (following the default setting in DomainBed (Gulrajani & Lopez-Paz, 2020)). Additional results regarding the impact of hyper-parameter choices and performance gain of FedBR on other datasets/settings/evaluation metrics can be found in Appendix C.

5.2. Numerical Results

The superior performance of FedBR over existing FL and DG algorithms.[5] In Table 1 and Figure 5, we show the performance and convergence curve of baseline methods as well as our proposed FedBR algorithm. When comparing different FL and DG algorithms, we discovered that: 1) FedBR performs best in all settings; 2) DG baselines only slightly outperform ERM, and some are even worse; 3) Regularizing local models to global models from prior rounds, such as Moon and Fedprox, does not result in positive outcomes.

Comparison with VHL. We vary the size of virtual data in VHL and compare it with our FedBR in Table 2 [6]: our communication-efficient FedBR only uses 32 pseudo-data and transfers pseudo-data once, while the communication-intensive VHL (Tang et al., 2022) requires the size of virtual data to be proportional to the number of classes and uses at least 2,000 virtual data (the authors suggest 2,000 for CIFAR10 and 20,000 for CIFAR100 respectively in the released official code, and we use the default value of hyper-parameters and implementation provided by the authors). We can find that 1) FedBR always outperforms VHL. 2) FedBR overcomes several shortcomings of VHL, e.g. the need for labeled virtual data and the large size of the virtual dataset.

5.3. Ablation Studies

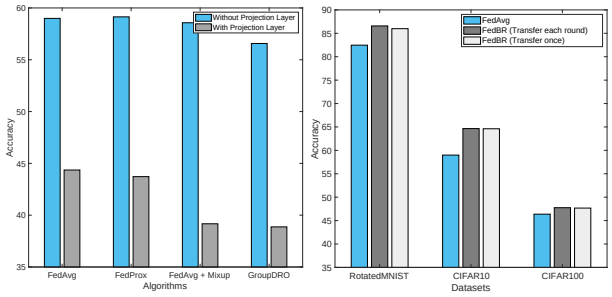
Effectiveness of the different components in FedBR. In Table 4, we show the improvements brought by different components of FedBR. In order to highlight the importance

[5] See CIFAR10 + ResNet18 results in Table 9 of Appendix C.

[6] The performance of FedBR is slightly different from results in Table 1 because we only use 32 pseudo-data here to make a fair comparison with VHL.

Table 4: **Ablation studies of FedBR** on the effects of two components. We show the performance of two components and remove the max step (Line 8 in Algorithm 1) of component 2. We split RotatedMNIST, CIFAR10, and CIFAR100 to 10 clients with $\alpha = 0.1$. We run 1000 communication rounds on RotatedMNIST and CIFAR10 for each algorithm and 800 communication rounds on CIFAR100. We report the mean of maximum (over rounds) 5 test accuracies and the number of communication rounds to reach the target accuracy.

Algorithm	RotatedMNIST (CNN)		CIFAR10 (VGG11)		CIFAR100 (CCT)	
	Acc (%)	Rounds for 80%	Acc (%)	Rounds for 55%	Acc (%)	Rounds for 43%
FedAvg	82.47	828 (1.0X)	58.99	736 (1.0X)	46.37	358 (1.0X)
Component 1	84.40	770 (1.1X)	64.32	**442 (1.7X)**	47.22	330 (1.1X)
+ min step	80.81	922 (0.9X)	62.98	562 (1.3X)	46.54	358 (1.0X)
Component 2	86.25	648 (1.3X)	63.44	483 (1.5X)	**47.78**	308 (1.2X)
+ w/o max step	81.24	926 (0.9X)	58.84	584 (1.3X)	43.50	512 (0.7X)
FedBR	**86.58**	**628 (1.3X)**	**64.65**	496 (1.5X)	**47.75**	**294 (1.2X)**



(a) w/ and w/o projection layer. (b) comm. of pseudo-data.

(c) choices of pseudo-data. (d) label distribution of RSM.

Figure 6: **Ablation studies of FedBR**, regarding the impact of the projection layer, the communication strategy of pseudo-data, and the choices of pseudo-data. In Figure 6(a), we show the performance of algorithms with/without the additional projection layer on the CIFAR10 dataset with the VGG11 model. In Figure 6(b), we show the performance of FedBR on RotatedMNIST, CIFAR10, and CIFAR100 datasets when only transferring pseudo-data once (at the beginning of training) or generating new pseudo-data each round. In Figure 6(c), we show the performance of FedBR using different types of pseudo-data. In Figure 6(d), we show the performance of FedBR when constructing RSM using data with balanced and unbalanced label distribution. Pseudo-data *transfer once* at the beginning of the training in Figure 6(c), and Figure 6(d).

of our two components, especially the max-step (c.f. Line 8 in Algorithm 1) in component 2, we first consider two components of FedBR individually, followed by removing the max-step. We find that: 1) Two components of FedBR have individual improvements compared with FedAvg, but the combined solution FedBR consistently achieves the best performance. 2) The projection layer is crucial.

Table 5: **Performance of algorithms with 100 clients.** We split CIFAR10 dataset into 100 clients with $\alpha = 0.1$. We run 1000 communication rounds for each algorithm on the VGG11 model and report the mean of the maximal 5 accuracies (over rounds) during training on test datasets.

Methods	FedAvg	FedDecorr	FedMix	FedProx	Mixup	VHL	FedBR
Acc	38.20	35.53	34.71	37.90	36.63	40.93	41.59

Table 6: **Performance of local model on balanced global test datasets.** We split CIFAR10 to 10 clients with $\alpha = 0.1$, and report the test accuracies achieved by the local models/aggregated models at the end of each communication round. For FedBR, pseudo-data only transfer once (32 pseudo-data).

Algorithm	FedAvg	FedDecorr	VHL	FedBR
Local Model Performance	21.01	21.18	32.81	21.83
Aggregated Model Performance	46.37	47.10	46.80	47.67

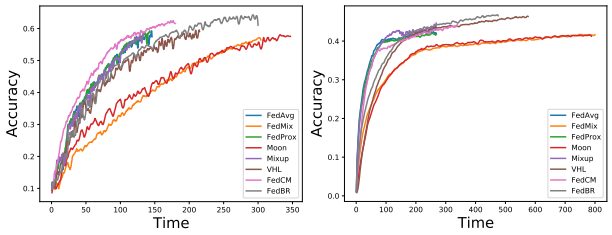
Table 7: **Parameter transmitted and mean simulation time in each round.** We split CIFAR10 and CIFAR100 to 10 clients with $\alpha = 0.1$. For FedBR, pseudo-data only transfer once (32 pseudo-data). The simulation time only includes the computation time per step, and do not includes the communication time. CIFAR100 experiments use Mixup as backbone.

CIFAR10 (VGG11)	FedAvg	Moon	VHL	FedCM	FedBR
Parameters (Millions)	9.2	9.7	9.2	18.4	9.7
Mean simulation time (s)	0.29	0.69	0.43	0.36	0.60
CIFAR100 (CCT)	FedAvg	Moon	VHL	FedCM	FedBR
Parameters (Millions)	22.4	22.6	22.4	44.8	22.6
Mean simulation time (s)	0.67	1.97	1.44	0.85	1.19

After removing projection layers, Component 2 of FedBR performs even worse than FedAvg; such insights may also explain the limitations of Moon (Li et al., 2021).

Performance of FedBR on CIFAR10 with different number of clients. In Table 5, we increase the number of clients to 100, and 10 clients are randomly chosen in each communication round. We can find that FedBR consistently outperform other methods.

Reducing the communication cost of FedBR. To reduce the communication overhead, we reduce the size of pseudo-data, and only transmit one mini-batch of pseudo-data (64



(a) CIFAR10 (b) CIFAR100

Figure 7: **Convergence curve w.r.t. simulation time.** We split CIFAR10 and CIFAR100 datasets to 10 clients, and report the mean accuracy on all local test datasets at each time slots. We use VGG11 for CIFAR10 experiments, and CCT for CIFAR100 experiments. CIFAR100 experiments use Mixup as backbone for all the algorithms. More details refer to Figure 16 of Appendix C.

for MNIST and 32 for others) once at the beginning of training. In Figure 6(b), we show the performance of FedBR when pseudo-data only transfer to clients at the beginning of the training (64 pseudo-data for RotatedMNIST, and 32 for CIFAR10 and CIFAR100). Results show that only transferring pseudo-data once can achieve comparable performance gain compared with transferring pseudo-data in each round. This indicates that the performance of FedBR will not drop even if we give a small number of pseudo-data.

Regarding privacy issues caused by RSM. Because RSM may have some privacy issues, we consider using Mixture to protect privacy. In Figure 6(c), we show the performance of FedBR with different types of pseudo-data (pseudo-data only transfer once at the beginning of training as in Figure 6(b)). Results show that: 1) FedBR consistently outperforms FedAvg on all types of pseudo-data. 2) When using **Mixture** as pseudo-data and setting $K = 0$ ((4)), FedBR still have a performance gain compared with FedAvg, and a more significant performance gain can be observed by setting $K = 1$.

Constructing pseudo-data by RSM using local data with unbalanced label distribution. In Figure 6(d), we construct the pseudo-data for FedBR using data with (1) balanced and (2) unbalanced label distributions. Results show that the performance of FedBR remained the same even when the data used to create the pseudo-data had an unbalanced label distribution.

Combining FedBR with other FL methods enhances performance. In Table 3, we combine FedBR with other SOTA FL algorithms, including FedNTD Lee et al. (2022), FedCM (Xu et al., 2021) and FedDecorr (Shi et al., 2023). Results demonstrate that FedBR significantly enhances the performance of these methods through simple integration.

Effectiveness of FedBR on reducing the local learning bias. To validate FedBR's ability to train unbiased local models, we save and assess the local models at the end of each communication round using balanced global test datasets. Results in Table 6 show that: 1) FedBR can achieve better performance than FedAvg without using the labeled global shared data, and the aggregated model matches and even surpasses VHL's performance 2) Local models of VHL perform better than other methods by using labeled global shared datasets to correct classification errors. It is natural for VHL to achieve better local performance as local datasets of VHL is relatively balanced.

Performance of FedBR regarding communication and computation costs. Using pseudo-data and an additional projection layer in FedBR increases computation and communication costs. We quantify this by reporting transmitted parameters and mean simulation time per round in Table 7, and display the convergence curve with respect to the simulation time in Figure 7. Results show that: 1) The computation time of FedBR is similar to that of other FL methods that adding regularization terms to overcome the local learning bias, such as VHL and Moon. 2) FedBR's communication cost remains minimal as it only introduce an additional small three-layer MLP projection layer, in contrast to the larger feature extractors found in modern deep neural networks.

6. Conclusion and Future works

We propose a new algorithm, FedBR, for Federated Learning that uses label-agnostic pseudo-data to improve performance on heterogeneous data. It has two key components and experiments show it significantly improves Federated Learning. Unlike previous methods, FedBR does not require labeled pseudo-data or a large pseudo-dataset, therefore reducing the communication costs.

However, FedBR requires additional computation as the algorithm needs additional forward propagation on pseudo-data, as well as the additional computation on the min-max optimization procedure. It would be interesting to explore ways to reduce this extra computation in the future.

Acknowledgement

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. This work is also supported in part by the National Natural Science Foundation of China (Grant No. 72171206, No. 71931003, No. 72061147004, No. 72192805 and No. 62001412), the National Key R&D Program of China (Grant No. 2018YFB1800800), the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS).

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization, 2019. URL https://arxiv.org/abs/1907.02893.

Bai, R., Bagchi, S., and Inouye, D. I. Benchmarking algorithms for domain generalization in federated learning, 2023. URL https://openreview.net/forum?id=IsCg7qoy8i9.

Chen, H.-Y. and Chao, W.-L. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Das, R., Acharya, A., Hashemi, A., Sanghavi, S., Dhillon, I. S., and Topcu, U. Faster non-convex federated learning via global and local momentum. *arXiv preprint arXiv:2012.04061*, 2020.

Deng, Y., Kamani, M. M., and Mahdavi, M. Distributionally robust federated averaging, 2021.

Duan, M., Liu, D., Chen, X., Tan, Y., Ren, J., Qiao, L., and Liang, L. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. *2019 IEEE 37th International Conference on Computer Design (ICCD)*, Nov 2019. doi: 10.1109/iccd46524.2019.00038. URL http://dx.doi.org/10.1109/ICCD46524.2019.00038.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. 2015. doi: 10.48550/ARXIV.1505.07818. URL https://arxiv.org/abs/1505.07818.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization, 2020. URL https://arxiv.org/abs/2007.01434.

Guo, Y., Lin, T., and Tang, X. Towards federated learning on time-evolving heterogeneous data. *arXiv preprint arXiv:2112.13246*, 2021.

Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., and Shi, H. Escaping the big data paradigm with compact transformers. 2021. URL https://arxiv.org/abs/2104.05704.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Jiang, L. and Lin, T. Test-time robust personalization for federated learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3aBuJEza5sq.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning, 2019. URL https://arxiv.org/abs/1910.06378.

Karimireddy, S. P., Jaggi, M., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex), 2020. URL https://arxiv.org/abs/2003.00688.

Lee, G., Jeong, M., Shin, Y., Bae, S., and Yun, S. Preservation of the global knowledge by not-true distillation in federated learning. In *36th Conference on Neural Information Processing Systems, NeurIPS 2022*. Advances in Neural Information Processing Systems (NeurIPS), 2022.

Li, D. and Wang, J. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization, 2017. URL https://arxiv.org/abs/1710.03463.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 5400–5409. Computer Vision Foundation / IEEE Computer Society, 2018a. doi: 10.1109/CVPR.2018.00566. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Li_Domain_Generalization_With_CVPR_2018_paper.html.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks, 2018b. URL https://arxiv.org/abs/1812.06127.

Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018c.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2351–2363. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=B1eyO1BFPr.

Luo, M., Chen, F., Hu, D., Zhang, Y., Liang, J., and Feng, J. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-efficient learning of deep networks from decentralized data. 2016. doi: 10.48550/ARXIV.1602.05629. URL https://arxiv.org/abs/1602.05629.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning, 2019.

Peng, X., Huang, Z., Zhu, Y., and Saenko, K. Federated adversarial domain adaptation, 2019. URL https://arxiv.org/abs/1911.02054.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2019. URL https://arxiv.org/abs/1911.08731.

Shen, Y., Du, J., Zhao, H., Zhang, B., Ji, Z., and Gao, M. Fedmm: Saddle point optimization for federated adversarial domain adaptation. *arXiv preprint arXiv:2110.08477*, 2021.

Shi, Y., Liang, J., Zhang, W., Tan, V. Y., and Bai, S. Towards understanding and mitigating dimensional collapse in heterogeneous federated learning. *International Conference on Learning Representations*, 2023.

Shin, M., Hwang, C., Kim, J., Park, J., Bennis, M., and Kim, S.-L. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning, 2020.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation, 2016. URL https://arxiv.org/abs/1607.01719.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Tang, Z., Zhang, Y., Shi, S., He, X., Han, B., and Chu, X. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. *arXiv preprint arXiv:2206.02465*, 2022.

Tuor, T., Wang, S., Ko, B. J., Liu, C., and Leung, K. K. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5020–5027. IEEE, 2021.

Wang, B., Li, G., Wu, C., Zhang, W., Zhou, J., and Wei, Y. A framework for self-supervised federated domain adaptation. *EURASIP Journal on Wireless Communications and Networking*, 2022(1):1–17, 2022.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020a.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020b.

Wu, S., Li, T., Charles, Z., Xiao, Y., Liu, Z., Xu, Z., and Smith, V. Motley: Benchmarking heterogeneity and personalization in federated learning. *arXiv preprint arXiv:2206.09262*, 2022.

Xu, J., Wang, S., Wang, L., and Yao, A. C.-C. Fedcm: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.

Yan, S., Song, H., Li, N., Zou, L., and Ren, L. Improve unsupervised domain adaptation with mixup training, 2020. URL https://arxiv.org/abs/2001.00677.

Yoon, J., Jeong, W., Lee, G., Yang, E., and Hwang, S. J. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021a.

Yoon, T., Shin, S., Hwang, S. J., and Yang, E. Fedmix: Approximation of mixup under mean augmented federated learning, 2021b.

Yoshida, N., Nishio, T., Morikura, M., Yamamoto, K., and Yonetani, R. Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks. *arXiv preprint arXiv:1905.07210*, 2019.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pp. 7252–7261. PMLR, 2019.

Contents of Appendix

A. Experiment Details

Framework and baseline algorithms. In addition to traditional FL methods, we aim to see if domain generalization (DG) methods can help increase model performance during FL training. Thus, we use the DomainBed benchmark (Gulrajani & Lopez-Paz, 2020), which contains a series of regularly used DG algorithms and datasets. The algorithms in DomainBed can be divided into three categories:

- **Infeasible methods:** Some algorithms can't be applied in FL scenarios due to the privacy concerns, for example, MLDG (Li et al., 2017), MMD (Li et al., 2018a), CORAL (Sun & Saenko, 2016), VREx (Krueger et al., 2020) that need features or data from each domain in each iteration.
- **Feasible methods (with limitations):** Some algorithms can be applied in FL scenarios with some limitations. For example, DANN (Ganin et al., 2015), CDANN (Li et al., 2018c) require knowing the number of domains/clients, which is impractical in the cross-device setting.
- **Feasible methods (without limitations):** Some algorithms can be directly applied in FL settings. For example, ERM, GroupDRO (Sagawa et al., 2019), Mixup (Yan et al., 2020), and IRM (Arjovsky et al., 2019).

We choose several common used DG algorithms that can easily be applied in Fl scenarios, including ERM, GroupDRO (Sagawa et al., 2019), Mixup (Yan et al., 2020), and DANN (Ganin et al., 2015). For FL baselines, we choose FedAvg (McMahan et al., 2016) (equal to ERM), Moon (Li et al., 2021), FedProx (Li et al., 2018b), SCAFFOLD (Karimireddy et al., 2019) and FedMix (Yoon et al., 2021b) which are most related to our proposed algorithms.

Notice that some existing works consider combining FL and domain generalization. For example, combining DRO with FL (Mohri et al., 2019; Deng et al., 2021), and combine MMD or DANN with FL (Peng et al., 2019; Wang et al., 2022; Shen et al., 2021). The natural idea of the former two DRO-based approaches is the same as our GroupDRO implementations, with some minor weight updates differences; the target of the later series of works that combine MMD or DANN is to train models to work well on unseen distributions, which is orthogonal with our consideration (overcome the local heterogeneity).To check the performance of this series of works, we choose to integrate FL and DANN into our environments.

Notice that we carefully tune all the baseline methods. The implementation detail of each algorithm is listed below:

- GroupDRO: The weight of each client is updated by $\boldsymbol{\omega}_i^{t+1} = \boldsymbol{\omega}_i^t \exp(0.01 l_i^t)$, where l_i^t is the loss value of client i at round t.
- Mixup: Local data is mixed by $\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$, and λ is sampled by $Beta(0.2, 0.2)$.
- DANN: Use a three-layer MLP as domain discriminator, where the width of MLP is 256. The weight of domain discriminate loss is tuned in $\{0.01, 0.1, 1\}$.
- FedProx: The weight of proximal term is tuned in $\{0.001, 0.01, 0.1\}$.
- Moon: The projection layer is a two-layer MLP, the MLP width is setting to 256, and the output dimension is 128. We tuned the weight of contrastive loss in $\{0.01, 0.1, 1, 10\}$.
- FedMix: The mixup weight λ used in FedMix is tuned in $\{0.01, 0.1, 0.2\}$, we construct 64 augmentation data in each local step for RotatedMNIST, and 32 samples for CIFAR10 and CIFAR100..
- VHL: We use the same setting as in the original paper, with the weight of augmentation classification loss $\alpha = 1.0$, and use the "proxy_align_loss" provided by the authors for feature alignment. Virtual data is generated by untrained style-GAN-v2, and we sample 2000 virtual data for CIFAR10 and RotatedMNIST; 20000 virtual data for CIFAR100 follow the default setting of the original work. To make a fair comparison, we sample 32 virtual samples in each local step for CIFAR10 and CIFAR100.
- FedNTD: We use the official code of FedNTD, set $\tau = 1.0$ as suggested in the original paper, and β is tuned in $\{1.0, 0.1\}$
- FedDecorr: We use the official code of FedDecorr, and set the weight of penalty term to 0.1.
- FedBR: We use a three-layer MLP as the projection layer, the MLP width is set to 256, and the output dimension is 128.

By default, we set $\tau_1 = \tau_2 = 2.0$, the weight of contrastive loss $\mu = 0.5$, and the weight of AugMean $\lambda = 1.0$ on MNIST and CIFAR100, $\lambda = 0.1$ on CIFAR10 and PACS. We sample 64 pseudo-data in each local step for RotatedMNIST and 32 samples for CIFAR10 and CIFAR100.

Datasets and Models. For datasets, we choose RotatedMNIST, CIFAR10, CIFAR100, and PACS. For RotatedMNIST, CIFAR10, and CIFAR100, we split the datasets following the idea introduced in (Yurochkin et al., 2019; Hsu et al., 2019; Reddi et al., 2021), where we leverage the Latent Dirichlet Allocation (LDA) to control the distribution drift with parameter α. Larger α indicates smaller non-iidness. We divided each environment into two clients for PACS, with the first client containing data from classes 0-3, and the second client containing data from classes 4-6.
Unless specially mentioned, we split RotatedMNIST, CIFAR10, and CIFAR100 to 10 clients and set $\alpha = 0.1$. For PACS, we have 8 clients instead. Notice that for each client of CIFAR10, we utilize a special transformation, i.e., rotation to the local data, to simulate the natural shift. In detail:

- RotatedMNIST: We first split MNIST by LDA using parameter $\alpha = 0.1$ to 10 clients, then for each client, we rotate the local data by $\{0, 15, 30, 45, 60, 75, 90, 105, 120, 135\}$.
- CIFAR10: We first split CIFAR10 by LDA using parameter $\alpha = 0.1$ to N clients. Then for each client, we sample $q \in \mathbb{R}^{10}$ from $Dir(1.0)$. For each image in local data, we sample an angle in $\{0, 15, 30, 45, 60, 75, 90, 105, 120, 135\}$ by probability q, and rotate the image by the angle.
- Clean CIFAR10: Unlike the previous setting, we do not rotate the samples in CIFAR10 (no inner-class non-iidness).
- CIFAR100: We split the CIFAR100 by LDA using parameter $\alpha = 0.1$, and transform the train data using RandomCrop, RandomHorizontalFlip, and normalization.

Each communication round includes 50 local iterations, with 1000 communication rounds for RotatedMNIST and CIFAR10, 800 communication rounds for CIFAR100, and 400 communication rounds for PACS. Notice that the number of communication rounds is carefully chosen, and the accuracy of all algorithms does not significantly improve after the given communication rounds.
The public data is chosen as RSM (Yoon et al., 2021b) by default, and we also provide results on other proxy datasets. We utilize a four-layer CNN for MNIST, VGG11 for CIFAR10 and PACS, and CCT (Hassani et al., 2021) (Compact Convolutional Transformer, cct_7_3x1_32_c100) for CIFAR100.
For each algorithm and dataset, we employ SGD as the optimizer, and set learning rate $lr = 0.001$ for MNIST, and $lr = 0.01$ for CIFAR10 , CIFAR100, and PACS. When using CCT and ResNet, we set momentum as 0.9. We set the same random seeds for all algorithms. We set local batch size to 64 for RotatedMNIST, and 32 for CIFAR10, CIFAR100, and PACS.

B. Details of Augmentation Data

We use the data augmentation framework the same as FedMix, as shown in Algorithm 2. For each local dataset, we upload the mean of each M samples to the server. The constructed augmentation data is close to random noise. As shown in Figure 8, we randomly choose one sample in the augmentation dataset of CIFAR10 dataset.

Algorithm 2 Construct Augmentation Data

Require: local Datasets $D_1, \ldots, D_N$, number of augmentation data for each client K, number of samples to construct one augmentation sample M.
Ensure: Augmentation Dataset D_p.
Initialize $D_p = \emptyset$.
for $i = 1, \ldots, N$ **do**
  for $k = 1, \ldots, K$ **do**
    Randomly sample $x_1, \ldots, x_M$ from D_i.
    $\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x_M$.
    $D_p = D_p \cup \{\bar{x}\}$

C. Additional Results

C.1. Results with Error Bar

In this section, we report the performance of our method FedAug and other baselines with an error bar to verify the performance gain of our proposed method.

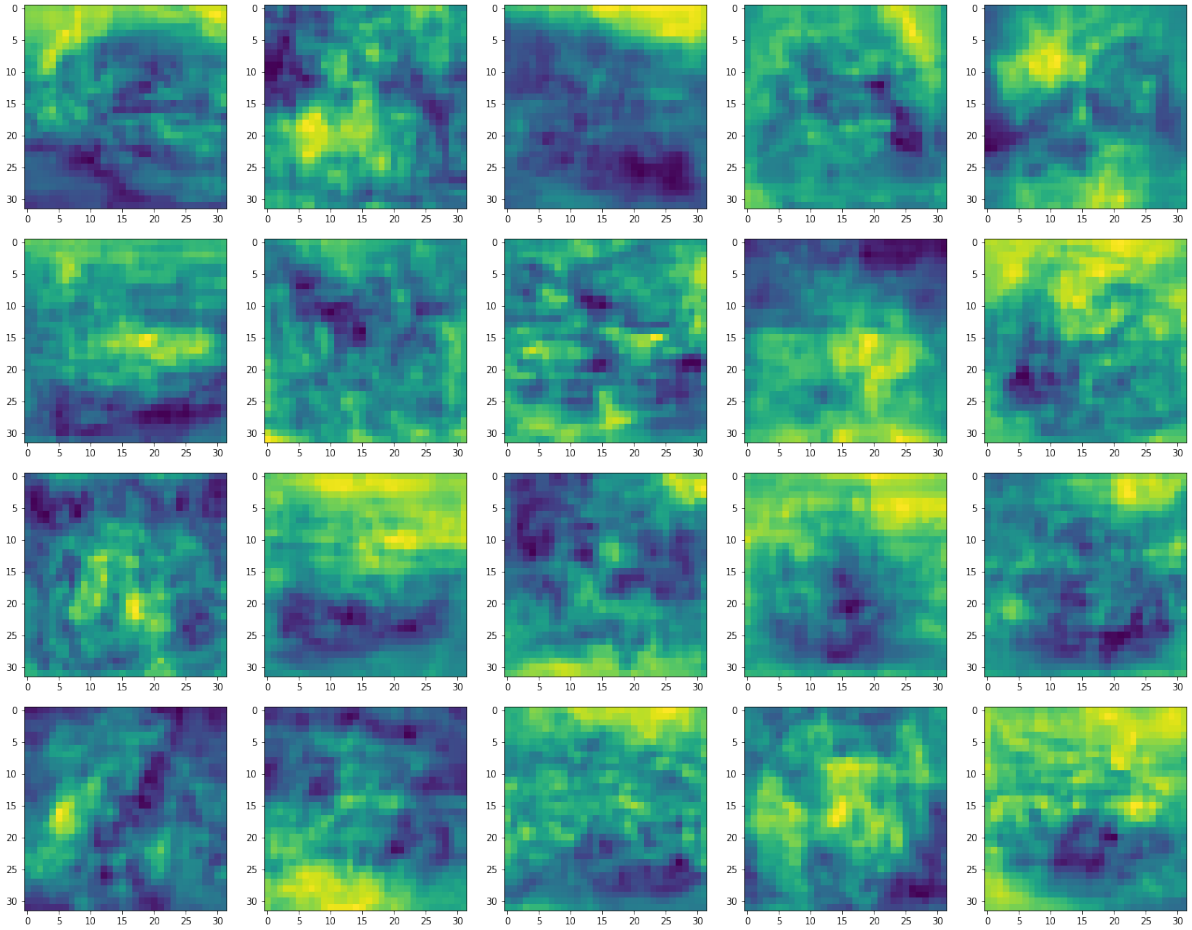
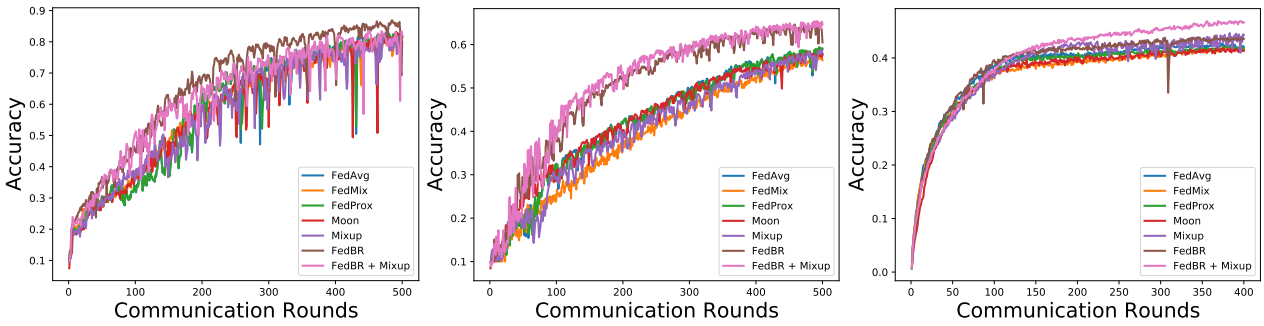


Figure 8: We show 20 augmentation data of CIFAR10 dataset here. Notice that the augmentation data is close to random noise and can not be classified as any class.



(a) Convergence curve on RotatedMNIST (b) Convergence curve on CIFAR10 (c) Convergence curve on CIFAR100

Figure 9: Convergence curve of algorithms on different datasets.

Algorithm 3 Construct Augmentation Data by Proxy Data

Require: Proxy Datasets D_{prox}, number of augmentation data K, number of samples to construct one augmentation sample M.
Ensure: Augmentation Dataset D_p.
1: Initialize $D_p = \emptyset$.
2: **for** $k = 1, \ldots, K$ **do**
3: Randomly sample $x_1, \ldots, x_M$ from D_{prox}.
4: $\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x_M$.
5: $D_p = D_p \cup \{\bar{x}\}$

Table 8: **Performance of algorithms with error bar.** All examined algorithms use FedAvg as the backbone. We run 1000 communication rounds on RotatedMNIST and CIFAR10 for each algorithm. For each algorithm, we run three different trials with different random seeds. For each trial, we report the mean of maximum 5 accuracies for test datasets and the number of communication rounds to reach the threshold accuracy.

Algorithm	RotatedMNIST		CIFAR10	
	Acc (%)	Rounds for 80%	Acc (%)	Rounds for 55%
ERM (FedAvg)	82.78 ± 0.38	821 (1.0X)	58.97 ± 0.30	742 (1.0X)
DANN	84.67 ± 0.46	754 (1.1X)	58.98 ± 0.61	747 (1.0X)
Mixup	82.38 ± 0.07	853 (1.0X)	58.32 ± 0.33	822 (0.9X)
GroupDRO	80.65 ± 0.53	929 (0.9X)	56.72 ± 0.26	840 (0.9X)
FedBR (Ours)	$\mathbf{87.05 \pm 0.44}$	**637 (1.3X)**	$\mathbf{64.62 \pm 0.32}$	**374 (2.0X)**

Table 9: **Performance of algorithms on CIFAR10.** We split CIFAR10 dataset to 10 clients with $\alpha = 0.1$, without additional rotation. For each algorithm, we run 1000 communication rounds on ResNet18 (with group normalization), and set local steps to 50. Note that we set momentum to 0.9 for ResNet18.

	FedAvg	FedProx	Moon	VHL	FedBR (ours)
Accuracy (ResNet18)	45.91	46.28	43.85	43.7	47.29

C.2. Ablation Study of FedBR

Values of τ_1 and τ_2 in Component 2. In this paragraph, we investigate how the value of τ_1 and τ_2 affect the performance of the second component of FedBR. In table 10, we show the results on Rotated-MNIST dataset with different weights τ_1 and τ_2. Results show that: 1) Setting $\tau_2 = 0$, which only minimizes the distance of global and local features, has significant performance gain compare with ERM. However, adding τ_2 can further improve the performance. 2) The best weight on Rotated-MNIST dataset is $\tau_1 = 2.0$ and $\tau_2 = 0.5$.

Table 10: **Performance of Component 2 of FedBR under different values of τ_1, τ_2.** We run 1000 communication rounds on RotatedMNIST dataset. For each setting, we run three different trials with different random seeds. For each trial, we report the mean of maximum 5 accuracies for test datasets and the number of communication rounds to reach the threshold accuracy.

τ_1	τ_2	Acc (%)	Rounds for 80%	Rounds for 85%
2.0	0.0	86.11 ± 0.77	746	933
2.0	0.1	86.22 ± 0.33	753	920
2.0	0.5	87.24 ± 0.50	647	851
2.0	1.0	86.25 ± 0.87	705	922
2.0	2.0	86.01 ± 0.33	680	932

Table 11: **Performance of FedBR under different values of τ_1, τ_2.** We run 1000 communication rounds on the CIFAR10 dataset. For each setting, we run three different trials with different random seeds. For each trial, we report the mean of maximum 5 accuracies for test datasets and the number of communication rounds to reach the threshold accuracy.

τ_1	τ_2	Acc (%)	Rounds for 55%	Rounds for 60%
2.0	0.0	64.05 ± 0.27	390	563
2.0	0.5	64.26 ± 0.47	382	585
2.0	1.0	64.77 ± 0.24	374	533
2.0	2.0	64.62 ± 0.32	374	541

Weights of the first component of FedBR. In this paragraph, we investigate how the weights of the first component of FedBR affect the performance of models in table 12.

Table 12: **Performance of component 1 under different weights.** We run 1000 communication rounds on the CIFAR10 dataset. For each setting, we run three different trials with different random seeds. For each trial, we report the mean of maximum 5 accuracies for test datasets and the number of communication rounds to reach the threshold accuracy. We use λ as the weight of the first component of FedBR.

λ	Acc (%)	Rounds for 55%	Rounds for 60%
0.1	64.12 ± 0.27	442	591
0.5	64.92 ± 0.46	385	536
1.0	64.50 ± 0.34	379	565

Domain robustness of FL and DG algorithms. We also hope that our method can increase the model's robustness because it expects to train client invariant features. Therefore, we calculate the worst accuracy on test datasets of all clients/domains and report the mean of each algorithm's top 5 worst accuracies in Table 14 to show the domain robustness of algorithms. We have the following findings: 1) FedBR significantly outperforms other approaches, and the improvements of FedBR over FedAvg become more significant than the mean accuracy in Table 1. FedBR has a role in learning a domain-invariant feature and improving robustness, as evidenced by this finding. 2) Under these settings, DG baselines outperform FedAvg. This finding demonstrates that the DG algorithms help to enhance domain robustness.

Table 13: **Performance of algorithms.** All examined algorithms use FedAvg as the backbone. We run 1000 communication rounds on RotatedMNIST and CIFAR10 for each algorithm, 800 communication rounds CIFAR100 and 400 communication rounds for PACS. We report the mean of maximum 5 accuracies for test datasets and the number of communication rounds to reach the final accuracy of ERM .

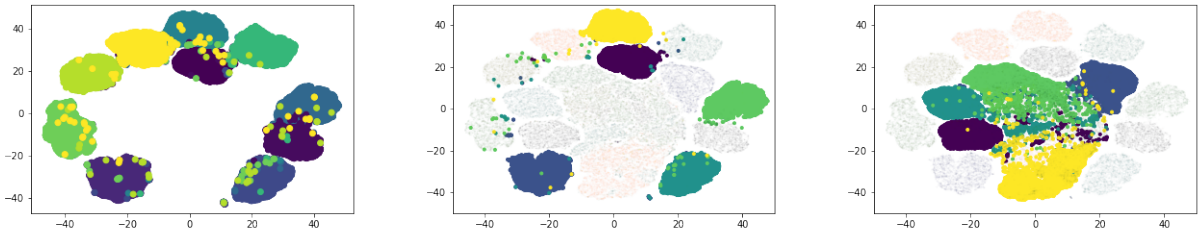
Algorithm	RotatedMNIST		CIFAR10		PACS	
	Acc (%)	Rounds (Speed up)	Acc (%)	Rounds (Speed up)	Acc (%)	Rounds (Speed up)
ERM (FedAvg)	82.47	828 (1.0X)	58.99	736 (1.0X)	64.03	168 (1.0X)
FedProx	82.32	824 (1.0X)	59.14	738 (1.0X)	65.10	168 (1.0X)
SCAFFOLD	82.49	814 (1.0X)	59.00	738 (1.0X)	64.49	168 (1.0X)
FedMix	81.33	902 (0.9X)	57.37	872 (0.8X)	62.14	228 (0.7X)
Moon	82.68	864 (0.9X)	58.23	820 (0.9X)	64.86	122 (1.4X)
DANN	84.83	743 (1.1X)	58.29	782 (0.9X)	64.97	109 (1.5X)
Mixup	82.56	840 (1.0X)	58.57	826 (0.9X)	64.36	210 (0.8X)
GroupDRO	80.23	910 (0.9X)	56.57	835 (0.9X)	64.40	170 (1.0X)
FedBR (Ours)	**86.58**	**628 (1.3X)**	**64.65**	**496 (1.5X)**	**65.63**	**100 (1.7X)**

Table 14: **Worst Case Performance of algorithms.** All examined algorithms use FedAvg as the backbone. We run 1000 communication rounds on RotatedMNIST and CIFAR10 for each algorithm, 800 rounds for CIFAR100, and 400 communication rounds for PACS. We calculate the worst accuracy for all clients in each round and report the mean of the top 5 worst accuracies for each method. Besides, we report the number of communication rounds to reach the final worst accuracy of FedAvg.

Algorithm	RotatedMNIST		CIFAR10		PACS	
	Acc (%)	Rounds (Speed up)	Acc (%)	Rounds (Speed up)	Acc (%)	Rounds (Speed up)
ERM (FedAvg)	66.60	816 (1.0X)	41.30	846 (1.0X)	42.79	170 (1.0X)
FedProx	65.88	780 (1.0X)	41.84	840 (1.0X)	42.82	170 (1.0X)
SCAFFOLD	66.72	804 (1.0X)	40.88	840 (1.0X)	41.63	170 (1.0X)
FedMix	60.52	910 (0.9X)	28.44	-	38.00	-
Moon	66.18	866 (0.9X)	40.34	908 (0.9X)	41.59	66 (2.6X)
DANN	67.85	753 (1.1X)	43.38	747 (1.1X)	40.51	59 (2.9X)
Mixup	66.25	836 (1.0X)	40.32	984 (0.9X)	41.89	252 (0.7X)
GroupDRO	68.53	**568 (1.4X)**	46.90	656 (1.3X)	43.18	246 (0.7X)
FedBR (Ours)	**77.13**	630 (1.3X)	**48.94**	**632 (1.3X)**	**43.99**	**58 (2.9X)**

C.3. T-SNE and Classcifier Outputs of Toy Examples

As the setting in Figure 2 and Figure 3, we investigate if the two components of FedBR will help for mitigating the proposed bias on feature and classifier. Figure 10 show the features after the second component of FedBR, which implies this component can significantly mitigate the proposed feature bias: 1) on the seen datasets, local features are close to global



(a) Global feature of X_1, $F_g(X_1)$ (b) Local feature of X_1, $F_1(X_1)$ (c) Local feature of X_2, $F_1(X_2)$

Figure 10: Features after the second component of FedBR.

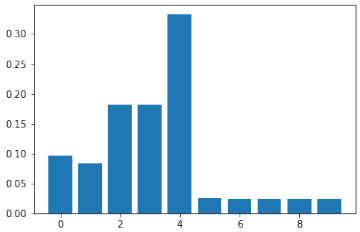
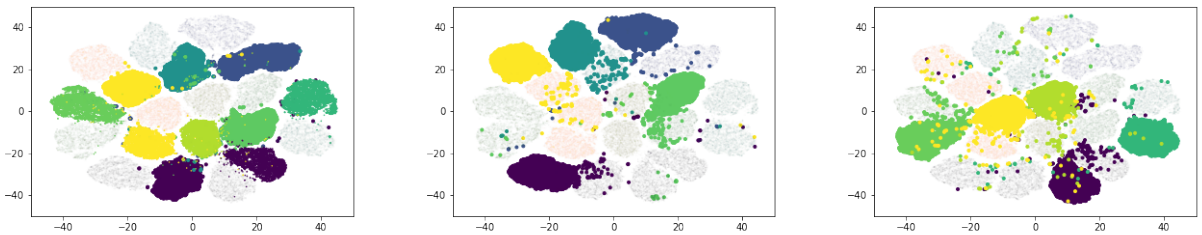


Figure 11: Classifier output after the first component of FedBR on unseen classes.

features. 2) on the unseen datasets, the local feature is far away from that of seen datasets. Figure 11 shows the output of the local classifier after the first component of FedBR on unseen classes. Notice that compared with Figure 3, the output is more balanced.

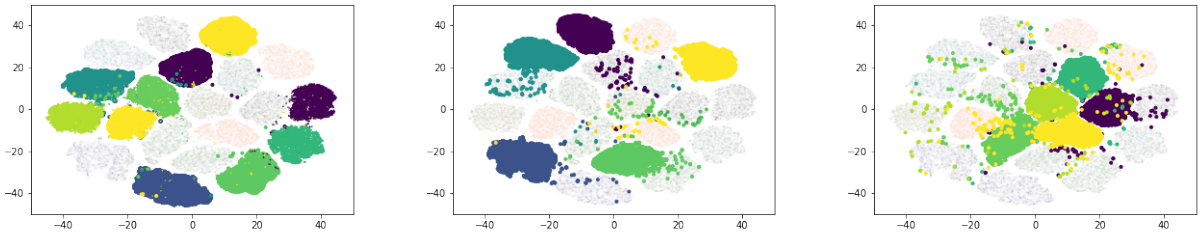
In Figure 12 and Figure 13, we show the local learning bias when local model has better feature initialization. We copy the feature extractor of global model to local models, and randomly initialize local classifiers. Results show that: 1) The drifts between global and local features are still significant even has a good feature initialization. 2) The local features of unseen data are less relevant to the local features of seen data compare with training from scratch. This indicates that such a problem will be mitigated after enough training rounds. 3) The drifts between global and local features increase as the number of local epochs increases.

We also investigate if our observation remains for different stages of global models. In this experiment, we use CIFAR10 dataset, and train global model for 1, 3, 10 epochs on the whole dataset to obtain 29.74%, 38.65%, 49.28% global accuracy, then we directly copy global models to clients (including classifier). We fine-tune the global models for 10 local epochs, results are shown in Figure 14. Results show that: For not well-trained global models, difference between global features on the same input and similarity between local features of different inputs are both significant.



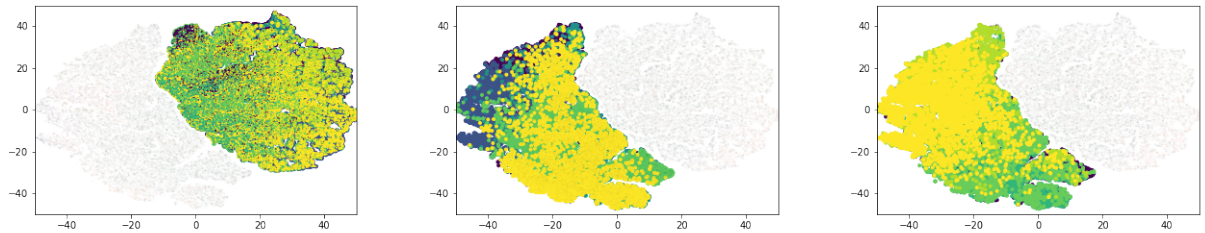
(a) Global feature of X_1, $F_g(X_1)$ (b) Fine-tuned Local feature of X_1, $F_1(X_1)$ (c) Fine-tuned Local feature of X_2, $F_1(X_2)$

Figure 12: Fine-tuned local features after 10 local epochs.

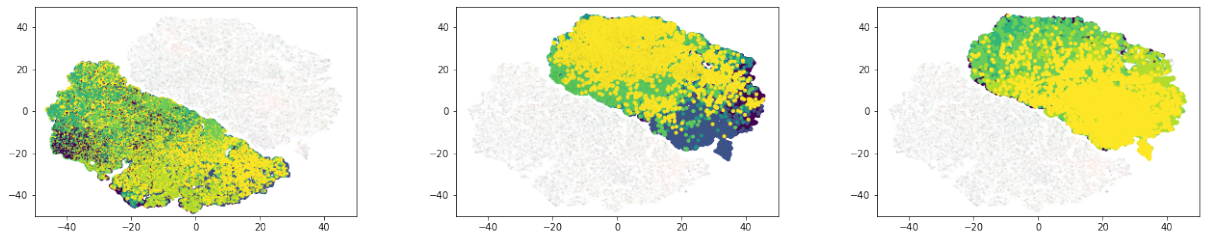


(a) Global feature of X_1, $F_g(X_1)$ (b) Fine-tuned Local feature of X_1, $F_1(X_1)$ (c) Fine-tuned Local feature of X_2, $F_1(X_2)$

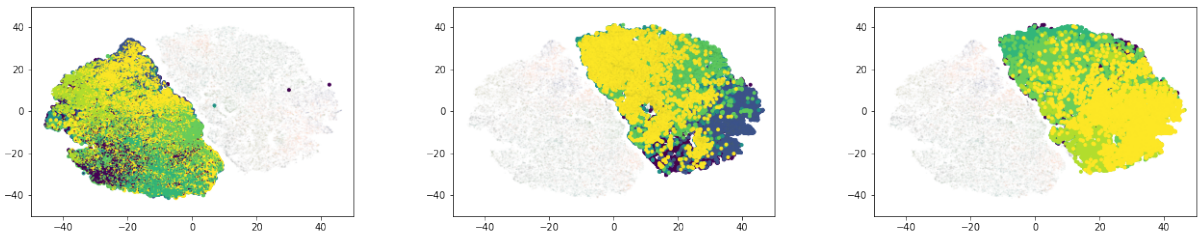
Figure 13: Fine-tuned local features after 20 local epochs.



(a) Global feature of X_1, $F_g(X_1)$ (b) Fine-tuned Local feature of X_1, $F_1(X_1)$ (c) Fine-tuned Local feature of X_2, $F_1(X_2)$

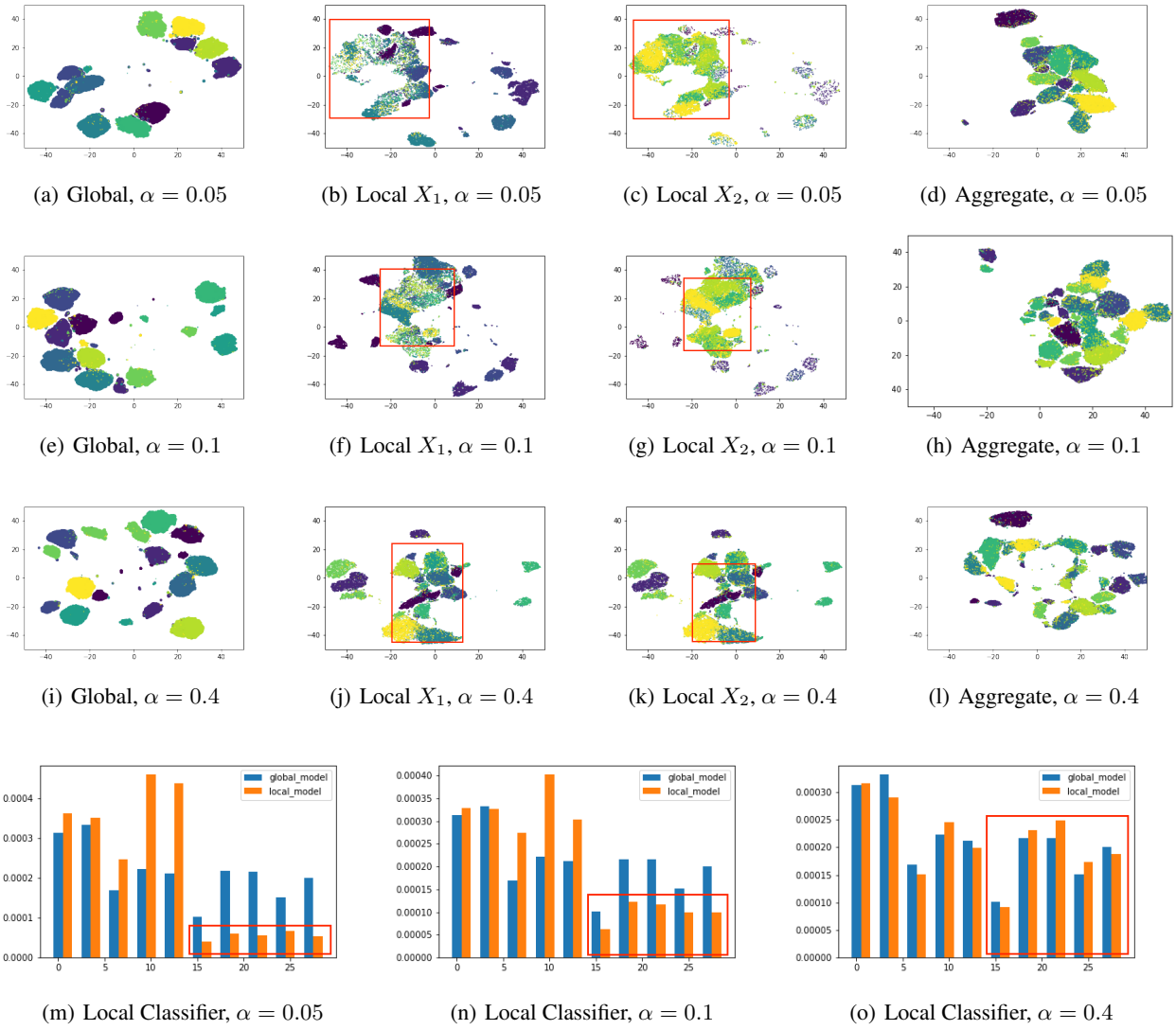


(d) Global feature of X_1, $F_g(X_1)$ (e) Fine-tuned Local feature of X_1, $F_1(X_1)$ (f) Fine-tuned Local feature of X_2, $F_1(X_2)$



(g) Global feature of X_1, $F_g(X_1)$ (h) Fine-tuned Local feature of X_1, $F_1(X_1)$ (i) Fine-tuned Local feature of X_2, $F_1(X_2)$

Figure 14: First train global model on the whole dataset for 1, 3, and 10 epoch (w.r.t. each row), then report local features after 10 local epochs.



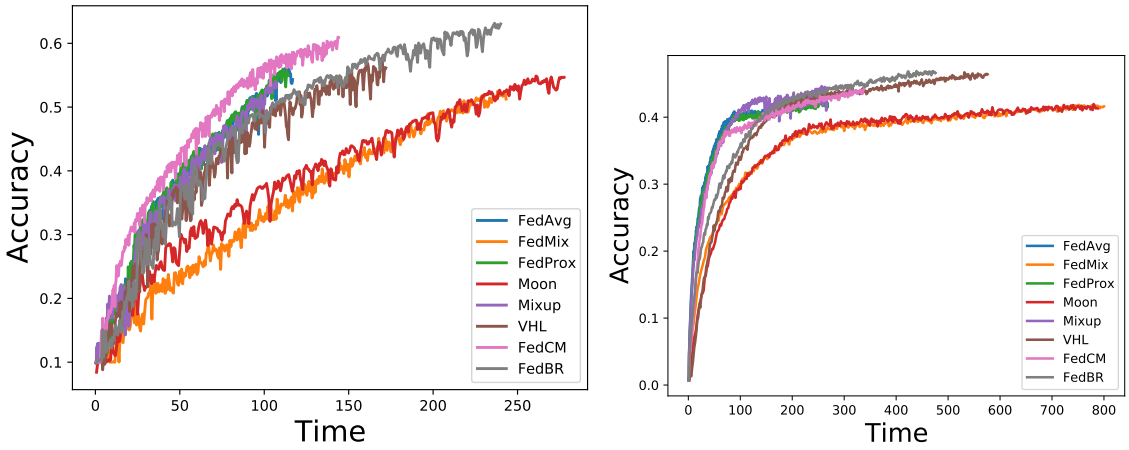
(a) Global, $\alpha = 0.05$ (b) Local X_1, $\alpha = 0.05$ (c) Local X_2, $\alpha = 0.05$ (d) Aggregate, $\alpha = 0.05$

(e) Global, $\alpha = 0.1$ (f) Local X_1, $\alpha = 0.1$ (g) Local X_2, $\alpha = 0.1$ (h) Aggregate, $\alpha = 0.1$

(i) Global, $\alpha = 0.4$ (j) Local X_1, $\alpha = 0.4$ (k) Local X_2, $\alpha = 0.4$ (l) Aggregate, $\alpha = 0.4$

(m) Local Classifier, $\alpha = 0.05$ (n) Local Classifier, $\alpha = 0.1$ (o) Local Classifier, $\alpha = 0.4$

Figure 15: **Illustration of our observation under mild split conditions:** We introduce a parameter $\alpha \in [0, 0.5]$ to control the level of non-i.i.d. of clients, where a larger α indicates less non-i.i.d., and $\alpha = 0.5$ indicates a balanced local distribution. We present the global feature, local feature on seen (X_1) and unseen (X_2) data, as well as the feature of the aggregated model. Additionally, we illustrate the output distribution of the global and local classifiers on the test data with a balanced label distribution.



(a) CIFAR10, VGG11 (b) CIFAR100, CCT

Figure 16: **Convergence curve w.r.t simulation time.** We present the convergence curves of various algorithms with respect to simulation time. Our results indicate that FedBR introduces additional computational burden compared to FedAvg and FedProx. However, the computational efficiency of FedBR is comparable to that of other regularization-based baselines.

C.4. T-SNE Results on Mild Conditions

We introduce a parameter $\alpha \in [0, 0.5]$ to control the level of non-i.i.d. of clients, where a larger α indicates less non-i.i.d., and $\alpha = 0.5$ indicates a balanced local distribution. Results are shown in Figure 15: 1) The local feature on the unseen data (Local X_2) still lacks a clear decision boundary, and the local features are close even for data from different classes. 2) The decision boundary of the aggregated model becomes clearer as α increases, supporting the necessity of reducing the local bias. 3) Our observation on the biased classifier still holds, where a smaller α leads to a more biased classifier output.