On Bridging Generic and Personalized Federated Learning

Hong-You Chen  
The Ohio State University, USA  
chen.9301@osu.edu

Wei-Lun Chao  
The Ohio State University, USA  
chao.209@osu.edu

Abstract

Federated learning is promising for its ability to collaboratively train models with multiple clients without accessing their data, but vulnerable when clients' data distributions diverge from each other. This divergence further leads to a dilemma: “Should we prioritize the learned model’s generic performance (for future use at the server) or its personalized performance (for each client)?” These two, seemingly competing goals have divided the community to focus on one or the other, yet in this paper we show that it is possible to approach both at the same time. Concretely, we propose a novel federated learning framework that explicitly decouples a model’s dual duties with two prediction tasks. On the one hand, we introduce a family of losses that are robust to non-identical class distributions, enabling clients to train a generic predictor with a consistent objective across them. On the other hand, we formulate the personalized predictor as a lightweight adaptive module that is learned to minimize each client’s empirical risk on top of the generic predictor. With this two-loss, two-predictor framework which we name Federated Robust Decoupling (FED-RoD), the learned model can simultaneously achieve state-of-the-art generic and personalized performance, essentially bridging the two tasks.

1 Introduction

Large-scale data are the driving forces of modern machine learning but come with the risk of data privacy. In many applications like health care, data are required to be kept separate to enforce data ownership and protection, hindering the collective wisdom (of data) for training strong models. Federated learning (FL), which aims to train a model with multiple data sources (i.e., clients) while keeping their data decentralized, has emerged as a popular paradigm to resolve these concerns [44].

The standard setup of FL seeks to train a single “global” model that can perform well on generic data distributions [44], e.g., the union of clients’ data. Since clients’ data are kept separate, mainstream algorithms like FEDAVG [69] take a multi-round approach shown in Figure 1. Within each round, the server first broadcasts the “global” model to the clients, who then independently update the model locally using their own (often limited) data. The server then aggregates the “local” models back into the “global” model and proceeds to the next round. This pipeline is shown promising if clients’ data are IID (i.e., with similar data and label distributions) [30, 84, 107, 109], which is, however, hard to meet in reality and thus results in a drastic performance drop [60, 106]. Instead of sticking to a single “global” model that features the generic performance, another setup of FL seeks to acknowledge the heterogeneity among clients by constructing a “personalized” model for each client that is tied to client’s personalized data [20, 32, 83]. This latter setup (usually called personalized FL) is shown to outperform the former (which we name generic FL) regarding the test accuracy of each client alone.

So far, these two seemingly contrasting FL setups are developed independently. In this paper, we however found that they can be approached simultaneously by generic FL algorithms like FEDAVG.

Preprint. Under review.
Concretely, algorithms designed for generic FL (G-FL) often discard the local models \( \{ w_m \} \) after training (see Figure 1). As a result, when they are evaluated in a personalized setting (P-FL), it is the global model \( \bar{w} \) being tested. Here, we found that if we instead keep \( \{ w_m \} \) and evaluate them in P-FL, they outperform nearly all the existing P-FL algorithms. Namely, personalized models seem to come for free from the local training step in generic FL.

At first glance, this may not be totally surprising since local training in G-FL algorithms is driven by the client’s empirical risk, which is what a personalized model seeks to optimize\(^1\). What really surprises us is that even without an explicit regularization term imposed by most P-FL algorithms \([20, 32, 83]\), the local models of G-FL algorithms can achieve better generalization performance. We conduct a detailed analysis and argue that the global aggregation step — taking average over model weights — indeed acts like a regularizer. Moreover, applying advanced G-FL algorithms \([1, 47, 59]\) to improve the G-FL accuracy seems to not hurt the “local” models’ P-FL accuracy.

Building on these observations, we dig deeper into generic FL. Specifically for classification, the non-IID clients can result from either non-identical class distributions or class-conditional data distributions, or both. One way to mitigate their influences is to make the local training objectives more aligned among clients. While this can hardly be achieved for the latter case without knowing clients’ data, we can do so for the former case by setting a consistent goal among clients — the learned local models should classify every class well, even if clients’ data have different class distributions.

![Figure 1: The multi-round generic FL pipeline (top).](image1)

We show that this can be realized by viewing each client’s local training as an independent class-balanced objective function. We validate \( F_{ED} \) will be shown in section 5, these class-balanced objective functions lead to much consistent local training among clients, making the resulting “global” model more robust to non-IID conditions.

The use of class-balanced objectives, nevertheless, degrades the local models’ P-FL performance. This is because the local models are no longer learned to optimize clients’ empirical risks. To address this issue, we propose a unifying framework for G-FL and P-FL which explicitly decouples a local model’s dual duties. Concretely, we introduce a lightweight personalized predictor (head) on top of the local model’s feature extractor, and train it with client’s empirical risk (see Figure 2). In other words, our framework involves (a) learning a single global model by optimizing local models with a class-balanced objective robust to non-identical class distributions, and (b) learning a personalized predictor for each local model by minimizing the empirical risk. We name this two-loss, two-predictor framework Federated Robust Decoupling (FED-ROD).

![Figure 2: Local training of FED-ROD.](image2)

Specifically for the personalized predictor, we propose to explicitly parameterize it with clients’ class distributions via a hypernetwork \([29]\). That is, we learn a shared meta-model that outputs personalized predictors for clients given their class distributions. This not only enables zero-shot model adaptation to new clients (without their data but class distributions), but also provides a better initialization to fine-tune the models for new clients given their data.

We validate FED-ROD on CIFAR \([52]\) and Fashion-MNIST \([94]\) under different non-IID settings. FED-ROD consistently outperforms existing generic and personalized FL algorithms in both setups. Moreover, FED-ROD is compatible with advanced generic FL algorithms like SCAFFOLD \([47]\) and FEEDYN \([1]\) to further improve their performance whenever non-identical class distributions occur.

\(^1\)However, the literature does not evaluate G-FL algorithms in the P-FL setup by their local models.
2 Related Work

2.1 Generic federated learning

FedAvg [69] is the standard algorithm, which involves multiple rounds of local training and global aggregation. Many works have studied its convergence [30, 48], robustness [7], communication [51, 75], especially for non-IID clients [59, 60, 106]. Many other works proposed to improve FedAvg. In terms of global aggregation, [89, 100] matched local model weights before averaging. [15, 33, 65, 108] replaced weight average by model ensemble and distillation. [36, 74] applied server momentum and adaptive optimization to improve the global model update. In terms of local training, [64, 67, 72, 99] improved the optimizer. To reduce local models’ drifts from the global model, [106] mixed client and server data in local training; FedProx [59], FedDANE [57], and FedDYN [1] employed regularization toward the global model; SCAFFOLD [46] MIME [47] leveraged control variates and/or server statistics to correct local gradients; [90, 96] modified the local model update rules. For most of them, the empirical risks on clients’ data are the major forces to drive local training.

We also aim to reduce local models’ drifts but via a different way. We directly bypass the empirical risks that reflect clients’ data distributions. Instead, we apply objective functions in class-imbalanced learning [34], which are designed to be robust to the change of class distributions. Our approach is different from [21, 92, 95], which monitored and resolved class imbalance from the server while we tackled it at the clients. Our approach is also different from agnostic FL [19, 70], whose local training is still built on empirical risk minimization. The closest to ours is [37], which used a traditional class-imbalanced treatment, re-weighting, to mitigate non-identical class distributions. We show that more advanced techniques can be applied to further improve the performance, especially under extreme non-IID conditions where re-weighting is less effective. Moreover, our method is compatible with existing efforts like FedDYN [1] and SCAFFOLD [46] to boost the generic performance.

2.2 Personalized federated learning

Personalized FL [53] learns a customized model for each client. Many approaches are based on multi-task learning (MTL) [22, 23, 39, 79, 104, 105] — leveraging the clients’ task relatedness to improve model generalizability. For instance, [83] encouraged related clients to learn similar models; [16, 20, 31, 32, 56, 58] regularized local models with a learnable global model, prior, or set of data logits. [4, 9, 61, 63] designed the model architecture to have both personalized (usually the feature extractor) and shareable components. [38, 103] constructed for each client an initialized model or regularizer based on learnable bases. Our approach is inspired by MTL as well but has several notable differences from existing works. First, we found that the global aggregation step in generic FL already serves as a strong regularizer. Second, instead of learning for each client a personalized feature extractor [9, 63] or an entire independent model that can operate alone [20, 32, 83], Fed-RoD shares a single feature extractor among all clients, inspired by invariant risk minimization [3, 5] and domain generalization [26, 71]. This reduces the total parameters to be learned and improves model’s generalizability. Compared to FedPER [4] which also learned a shared feature extractor, Fed-RoD simultaneously outputs a single, strong global model to excel in the generic FL setup.

Some other approaches are based on mixture models. [2, 18, 68, 73, 101] (separately) learned global and personalized models and performed a mixture of them for prediction. [76] learned a set of expert models and used them to construct personalized models. Meta-learning is also applied to learn a good initialized model that can be adapted to each client with a few steps of local training [14, 24, 42, 50].

Instead of designing specific algorithms for personalized FL, [91, 98] showed that performing post-processing (e.g., fine-tuning) to a generic FL model (e.g., \( \bar{w} \) learned by FedAvg) already leads to promising personalized accuracy. In this work, we further showed that, the local models \( w_m \), learned in FedAvg and other generic FL algorithms are indeed strong personalized models.

We note that, while many personalized FL algorithms also produce a global model, it is mainly used to regularize or construct personalized models but not for evaluation in the generic setup. In contrast, we learn models to excel in both the setups via a single framework without sacrificing either of them.

A concurrent work pFedHN [80] also applies hypernetworks [29] but in a very different way from Fed-RoD. Specifically, pFedHN learns a hypernetwork at the server to aggregate clients’ model updates and produce their entire models for the next round. In contrast, we learn the hypernetwork locally to construct the personalized predictors, not the entire models, for fast adaptation to clients.
3 Personalized Models Emerge from Generic Federated Learning

In this section, we show that personalized FL (P-FL) models emerge from the training process of generic FL (G-FL) algorithms. To begin with, we review representative G-FL and P-FL algorithms.

3.1 Background

Generic federated learning. In a generic FL setting with $M$ clients, where each client has a data set $\mathcal{D}_m = \{(x_i, y_i)\}_{i=1}^{D_m}$, the optimization problem to solve can be formulated as

$$\min_w \mathcal{L}(w) = \sum_{m=1}^{M} \frac{|\mathcal{D}_m|}{|\mathcal{D}|} \mathcal{L}_m(w), \quad \text{where} \quad \mathcal{L}_m(w) = \frac{1}{|\mathcal{D}_m|} \sum_i \ell(x_i, y_i; w).$$

Here, $w$ is the model parameter; $\mathcal{D} = \bigcup_m \mathcal{D}_m$ is the aggregated data set from all clients; $\mathcal{L}_m(w)$ is the empirical risk computed from client $m$’s data; $\ell$ is a loss function applied to each data instance.

Federated averaging (FEDAVG). As clients’ data are separate, Equation 1 cannot be solved directly. A standard way to relax it is FEDAVG [69], which iterates between two steps, local training and global aggregation, for multiple communication rounds

$$\textbf{Local: } w_m = \arg \min_w \mathcal{L}_m(w), \text{ initialized with } \bar{w}; \quad \textbf{Global: } \bar{w} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \frac{|\mathcal{D}_m|}{|\mathcal{D}|} w_m.$$  \hspace{1cm} (2)

The local training is performed at all (or part of) the clients in parallel, usually with multiple epochs of SGD to output the local model $w_m$. The global aggregation is by taking element-wise average over model weights. Since local training is driven by clients’ empirical risks, when clients’ data are non-IID, $w_m$ would drift away from each other, making $\bar{w}$ deviate from the solution of Equation 1.

Personalized federated learning. Personalized FL learns for each client $m$ a model $w_m$, whose goal is to perform well on client $m$’s data. While there is no agreed objective function so far, many existing works [16, 20, 31, 32, 56, 58, 83] define the optimization problems similar to the follows

$$\min_{\{(\Omega, w_1, \ldots, w_M)\}} \sum_{m=1}^{M} \frac{|\mathcal{D}_m|}{|\mathcal{D}|} \mathcal{L}_m(w_m) + \mathcal{R}(\Omega, w_1, \ldots, w_M),$$

(3)

where $\mathcal{R}$ is a regularizer; $\Omega$ is introduced to relate clients. The regularizer is imposed to prevent $w_m$ from over-fitting client $m$’s limited data. Unlike Equation 1, Equation 3 directly seeks to minimize each client’s empirical risk (plus a regularization term) by the corresponding personalized model $w_m$.

In practice, personalized FL algorithms often run iteratively between the local and global steps as well, so as to update $\Omega$ according to clients’ models. One example is to define $\Omega$ as a global model [20, 31, 32, 58], e.g., by taking average over clients’ models, and apply an $L_2$ regularizer between $\Omega$ and each $w_m$. The corresponding local training step thus could generally be formulated as

$$\textbf{Local: } w_m^{(t+1)} = \arg \min_w \mathcal{L}_m(w) + \frac{\lambda}{2} \|w - \Omega\|_2^2, \text{ initialized with } w_m^{(t)},$$

(4)

where $w_m^{(t)}$ denotes the local model after the $t$-th round; $\lambda$ is the regularization coefficient. It is worth noting that unlike Equation 2, $w$ in Equation 4 is initialized by $w_m^{(t)}$, not by $\Omega$ (or $\bar{w}$).

Terminology. Let us clarify the concepts of “global” vs. “local” models, and “generic” vs. “personalized” models. The former corresponds to the $\textbf{training}$ phase: local models are the ones after every round of local training, which are then aggregated into the global model at the server (Equation 2). The latter corresponds to the $\textbf{testing}$ phase: the generic model is used at the server for generic future test data, while personalized models are specifically used for each client’s test data.

3.2 Local models of generic FL algorithms are strong personalized models

Building upon the aforementioned concepts, we investigate the literature and found that when generic FL algorithms are evaluated in the P-FL setup, it is their global models being tested. In contrast, when personalized FL algorithms are applied, it is their local models (e.g., Equation 4) being tested. This discrepancy motivates us to instead evaluate generic FL algorithms using their local models.
Figure 1 summarizes the results (see section 5 for details). Using local models of FEDAVG (i.e., Equation 2) notably outperforms using its global model in the P-FL setup. At first glance, this may not be surprising, as local training in FEDAVG is driven by clients’ empirical risks. What really surprises us, as will be seen in section 5, is that FEDAVG’s local models outperform most of the existing personalized FL algorithms, even if no explicit regularization is imposed in Equation 2.

3.3 Initialization with weight average is a strong regularizer

For a further understanding, we plot FEDAVG local models’ accuracy on clients’ training and test data. We do so also for a state-of-the-art personalized algorithm DITTO [58], whose local training step for personalized models is similar to Equation 4. As shown in Figure 3, FEDAVG has a lower training but higher test accuracy, implying that FEDAVG’s local training is more regularized than Equation 4.

We attribute this effect to the initialization in Equation 2. Specifically, by initializing $\bar{w}$ with $w$, we essentially impose an $L_2$ regularizer $\frac{\lambda}{2}||w - \bar{w}||_2^2$ with $\lambda \to \infty$ in the beginning of each round of local training, followed by resetting $\lambda$ to be 0. We found that this implicit regularization leads to a smaller value of $||w - \bar{w}||_2^2$ in the end of each local training round, compared to Equation 4. Due to the page limit, we leave additional analyses in the supplementary material. We note that, advanced generic FL algorithms like SCAFFOLD [46] and FEDDYN [1] still apply this initialization and learn with the empirical risk during local training. Thus, their local models are strong personalized models as well.

4 Our Framework: Federated Robust Decoupling (FED-ROD)

The fact that personalized models emerge from generic FL algorithms motivate us to focus more on how to improve the latter, especially when clients have non-IID data distributions.

4.1 Improving the generic FL performance with Balanced Risk Minimization (BRM)

We first analyze what factors may lead to non-IID conditions. Suppose the data instance $(x, y)$ of client $m$ is sampled IID from a client-specific joint distribution $P_m(x, y) = P_m(x|y)P_m(y)$, then the non-IID distributions among clients can result from different $P_m(x|y)$, $P_m(y)$, or both. We note that, both of the differences in $P_m(x|y)$ and $P_m(y)$ can make $L_m(w)$ deviate from $L(w)$ in Equation 1, which is the main cause of degradation in generic FL [59, 60, 106].

One way to mitigate the influence of non-IID data is to make $L_m(w)$ align with each other. This can be challenging to achieve if clients have different $P_m(x|y)$: without knowing clients’ data\(^2\), it is hard to design such an aligned $L_m(w)$. However, when clients have different $P_m(y)$\(^3\), i.e., different and hence imbalanced class distributions, we can indeed design a consistent local training objective by setting a shared goal for the clients — the learned local models should classify all the classes well. It is worth noting that setting such a goal does not require every client to know others’ data.

Learning a classifier to perform well on all classes irrespective of the training class distribution is the main focus of class-imbalanced learning [40, 41, 43]. We therefore propose to treat each client’s local training as a class-imbalanced learning problem and leverage techniques developed in this sub-field. Re-weighting and re-sampling [8] are the most fundamental techniques. Denote by $N_{m, c}$ the number of training instances of class $c$ for client $m$, these techniques adjust $L_m(w)$ in Equation 1 into

$$L_m^{BR}(w) \propto \sum_i q_{yi} \ell(x_i, y_i; w),$$  \quad \text{where } q_{yi} = \text{usually set as } \frac{1}{N_{m,y_i}} \text{ or } \frac{1}{\sqrt{N_{m,y_i}}}, \quad (5)$$

Namely, they mitigate the influence of $P_m(y)$ by turning the empirical risk $L_m$ into a balanced risk $L_m^{BR}$, such that every client solves a more consistent objective that is robust to the class distributions. Recently, many class-imbalanced works proposed to replace the instance loss $\ell$ (e.g., cross entropy)

\(^2\)Clients having different $P_m(x|y)$ is related to domain adaptation [6, 25, 27, 66, 102] and generalization [26, 71], which however require knowing the distributions of all (or part of) the clients for algorithm design.

\(^3\)This is indeed the main cause of non-IID data distributions in the literature of FL [36, 37].
with a class-balanced loss [10, 45, 49, 77, 97], showing more promising results than re-weighting or re-sampling. We can also define $\mathcal{L}_m^{BR}$ using these losses, e.g., the balanced softmax (BSM) loss [77]

$$
\mathcal{L}_m^{BR}(w) \propto \sum_i^{\text{BSM}} (x_i, y_i; w), \quad \ell_{\text{BSM}}(x, y; w) = -\log \frac{N_{m,y} \exp(g_y(x; w))}{\sum_{c \in \mathcal{C}} N_{m,c} \exp(g_c(x; w))}.
$$

Here, $g_c(x; w)$ is the logit for class $c$, $\mathcal{C}$ is the label space, and $\gamma$ is a hyper-parameter. The BSM loss is an unbiased extension of softmax to accommodate the class distribution shift between training and testing. It encourages a minor-class instance to claim a larger logit in testing. It encourages a minor-class instance to claim a larger logit

In this paper, we take advantage of these existing efforts by replacing the empirical risk $\mathcal{L}_m$ in Equation 2 with a balanced risk $\mathcal{L}_m^{BR}$, which either takes the form of Equation 5 or applies a class-balanced loss (e.g., Equation 6), or both. We note that, a variant of Equation 5 has been used in [37]. However, our experiments show that it is less effective than class-balanced losses in extreme non-iid cases. Interestingly, we found that $\mathcal{L}_m$ can seamlessly be incorporated into advanced FL algorithms like FedDYN [1]. This is because these algorithms are agnostic to the local objectives being used.

4.2 Local training and local model decoupling with ERM and BRM

The use of balanced risk $\mathcal{L}_m^{BR}$ in local training notably improves the resulting global model $\bar{w}$’s generic performance, as will be seen in section 5. Nevertheless, it inevitably hurts the local models $w_m$’s personalized performance, since it is no longer optimized towards client’s empirical risk $\mathcal{L}_m$.

To address these contrasting pursuits of generic and personalized FL, we propose a unifying FL framework named **Federated Robust Decoupling (Fed-ROD)**, which **decouples the dual duties of local models** by learning two predictors on top of a shared feature extractor: one trained with empirical risk minimization (ERM) for personalized FL (P-FL) and the other with balanced risk minimization (BRM) for generic FL (G-FL). Figure 4 (c-d) illustrates the model and local training objective of Fed-ROD. The overall training process of Fed-ROD follows FedAvg, iterating between local training and global aggregation. As mentioned in subsection 4.1, other generic FL algorithms [1, 47, 59] can easily be applied to the BRM branch to further improve the generic performance. Without loss of generality, we focus on the basic version built upon FedAvg. We start with the model in Figure 4 (c).

**Notations.** We denote by $f(x; \theta)$ the shared feature extractor parameterized by $\theta$, whose output is $z$. We denote by $h^G(z; \psi)$ and $h^P(z; \phi_m)$ the generic and personalized prediction heads parameterized by $\psi$ and $\phi_m$, respectively; both are fully-connected (FC) layers. In short, our generic model is parameterized by $\{\theta, \psi\}$; our personalized model for client $m$ is parameterized by $\{\theta, \psi, \phi_m\}$.
Predictions. For generic prediction, we perform \( z = f(x; \theta) \), followed by \( \hat{y}_G = h^{G}(z; \psi) \). For personalized prediction, we perform \( f(x; \theta) \), followed by \( \hat{y}_P = h^{G}(z; \psi) + h^{P}(z; \phi_m) \). That is, \( h^{P} \) is an add-on to \( h^{G} \), providing personalized information that is not captured by the generic head.

The overall objective. 

Fed-ROD learns the generic model with the balanced risk \( L^{BR}_m \) and the personalized predictor with the empirical risk \( L_m \). That is, different from Equation 1, Fed-ROD aims to solve the following two optimization problems simultaneously

\[
\min_{\theta, \psi} \mathcal{L} (\{ \theta, \psi \}) = \sum_{m=1}^{M} \left[ \frac{|D_m|}{|D|} L^{BR}_m (\{ \theta, \psi \}) \right] \quad \text{and} \quad \min_{\phi_m} \mathcal{L}_m (\{ \theta, \psi, \phi_m \}), \forall m \in [M]. \tag{7}
\]

We note that, \( L_m \) is only used to learn the personalized head parameterized by \( \phi_m \).

Learning. Equation 7 cannot be solved directly in federated learning, so Fed-ROD follows FedAvg to learn iteratively between the local training and global aggregation steps

Local: \( \theta^*_m, \psi^*_m = \arg \min_{\theta, \psi} \mathcal{L}^{BR}_m (\{ \theta, \psi \}), \quad \text{initialized with } \bar{\theta}, \bar{\psi}, \tag{8} \)

\( \phi^*_m = \arg \min_{\phi_m} \mathcal{L}_m (\{ \theta, \psi, \phi_m \}), \quad \text{initialized with } \phi^*_m \tag{9} \)

Global: \( \theta \leftarrow \sum_{m=1}^{M} \left[ \frac{|D_m|}{|D|} \theta^*_m \right], \quad \psi \leftarrow \sum_{m=1}^{M} \left[ \frac{|D_m|}{|D|} \psi^*_m \right], \tag{10} \)

where \( \phi^*_m \) is learned from the previous round, similar to \( w_m^{(t)} \) in Equation 4. That is, the personalized head will not be averaged globally but kept locally. In our implementation, Equation 8 and Equation 9 are solved simultaneously via SGD, and we do not derive gradients w.r.t. \( \theta \) and \( \psi \) from \( \mathcal{L}_m (\{ \theta, \psi, \phi_m \}) \). The \( \theta \) and \( \psi \) in Equation 9 thus come dynamically from the SGD updates of Equation 8. In other words, Equation 9 is not merely fine-tuning on top of the generic model. In the end of federated learning, we will obtain \( \theta, \psi, \) and \( \{ \phi^*_m \}_{m=1}^{M} \) for generic and personalized predictions. Please be referred to the supplementary material for the pseudocode.

4.3 Adaptive personalized predictors via hypernetworks

In subsection 4.2, the parameter \( \phi_m \) of the personalized predictor is learned independently for each client and never shared across clients. In other words, for a new client not involved in the training phase, Fed-ROD can only offer the global model for generic prediction. In this subsection, we investigate learning a shared personalized predictor that can adapt to new clients. Concretely, we propose to learn a meta-model which can generate \( \phi_m \) for a client given the client’s class distribution. We denote by \( H^{P}(a_m; \nu) \) the meta-model parameterized by \( \nu \), whose output is \( \phi_m \). Here, \( a_m \in \mathbb{R}^{|C|} \) is the \(|C|\)-dimensional vector that records the class distribution of client \( m \); i.e., the \( c \)-th dimension \( a_m[c] = \sum_{n \in c} r_m, n \in c \). Accordingly, the local training step of \( \phi_m \) in Equation 9 is replaced by

Local: \( \nu^*_m = \arg \min_{\nu} \mathcal{L}_m (\{ \theta, \psi; \nu \}), \quad \text{initialized with } \bar{\nu}; \quad \text{Global: } \nu \leftarrow \sum_{m=1}^{M} \left[ \frac{|D_m|}{|D|} \nu^*_m \right]. \tag{11} \)

We implement \( H^{P} \) by a lightweight hypernetwork [29] with two fully-connected layers. With the learned \( \nu \), the meta-model \( H^{P} \) can locally generate \( \phi_m \) based on \( a_m \), making it adaptive to new clients simply by class distributions. The parameter \( \phi_m \) can be further updated using clients’ data.

We name this version Fed-ROD (hyper); the previous one, Fed-ROD (linear). Please see Figure 4 (c-d) for an illustration. We include more details in the supplementary material.

4.4 Extension with meta-learning

Zhao et al. [106] proposed to mitigate the local model drifts by sending a small labeled data set of the server to the clients. Fed-ROD can benefit from such a setting in local training, via meta-learning the hyper-parameters of \( L^{BR}_m \) [78, 81]. Please see the supplementary material for more details.

5 Experiment

Datasets, models, and settings. We use CIFAR-10 [52] and Fashion-MNIST (FMNIST) [94]. CIFAR-10 has 50K training and 10K test images from 10 classes. FMNIST is similar to MNIST [55], i.e., 60K training and 10K test images from 10 classes, but is more challenging in its contents.
Table 1: Main results in G-FL accuracy and P-FL accuracy (%). *: methods with no G-FL models and we combine their P-FL models. §: official implementation. Gray rows: meta-learning with 100 labeled server data.

| Dataset         | FMNIST | CIFAR-10 |
|-----------------|--------|----------|
| **Test Set**    |        |          |
| Non-IID         | Dir(0.1) | Dir(0.3) | Dir(0.1) | Dir(0.3) |
| G-FL            | GM | GM | GM | GM | GM | GM | GM | GM | GM |
| P-FL            | GM | GM | GM | GM | GM | GM | GM | GM | GM |
| **Method / Model** |        |          |          |          |          |          |          |          |          |
| FEDAVG [69]     | 81.1  | 81.0  | 91.5  | 83.4  | 83.2  | 90.5  | 57.6  | 57.1  | 90.5  | 68.6  | 69.4  | 85.1 |
| FedProx [59]    | 82.2  | 82.3  | 91.4  | 84.5  | 84.5  | 89.7  | 58.7  | 58.9  | 89.7  | 69.9  | 69.8  | 84.7 |
| SCAFFOLD [47]   | 83.1  | 83.0  | 89.0  | 85.1  | 85.0  | 90.4  | 61.3  | 60.8  | 90.1  | 71.1  | 71.5  | 84.8 |
| FedDyn [1]§     | 83.2  | 83.2  | 90.7  | 86.1  | 86.1  | 91.5  | 63.4  | 63.9  | 92.4  | 72.5  | 73.2  | 85.4 |
| MOCHA [83]¶     | 36.1  | 36.0  | 87.3  | 53.1  | 53.4  | 78.3  | 12.1  | 12.7  | 90.6  | 13.5  | 13.7  | 80.2 |
| LG-FEDAVG [63]¶ | 54.8  | 54.5  | 89.5  | 66.8  | 66.8  | 84.4  | 29.5  | 28.8  | 90.9  | 46.7  | 46.2  | 82.4 |
| FedPer [4]¶     | 74.5  | 74.4  | 91.3  | 79.9  | 79.9  | 90.4  | 50.4  | 50.2  | 89.9  | 64.4  | 64.5  | 84.9 |
| PER-FEDAVG [24] | 80.5  | 82.8  | 84.1  | 86.7  | 86.7  | 90.6  | 84.5  | 84.5  | 90.6  | 82.7  | 82.7  | 88.7 |
| PFedME [20]§    | 76.7  | 76.7  | 83.4  | 79.0  | 79.0  | 83.4  | 50.6  | 50.7  | 76.6  | 62.1  | 61.7  | 70.5 |
| DITTO [55]      | 81.5  | 81.5  | 89.4  | 83.3  | 83.2  | 90.1  | 58.1  | 58.3  | 86.8  | 69.7  | 69.8  | 81.5 |
| FEDFOMO [103]¶  | 34.5  | 34.3  | 90.0  | 70.1  | 69.9  | 89.6  | 30.5  | 31.2  | 90.5  | 45.3  | 45.1  | 83.4 |

*We use a ConvNet [55] similar to [1, 69, 86]. It contains 2 Conv layers and 2 FC layers. Please see the supplementary material for details. Centralized training leads to 85.4% and 93.5% accuracy on CIFAR-10 and FMNIST, respectively. To simulate the non-IID distributions of clients, we follow [36] to create a heterogeneous partition for $M$ clients on 10 classes. An $M$-dimensional vector $q_c$ is drawn from $\text{Dir}(\alpha)$ for class $c$, and we assign data to client $m$ proportionally to $q_c[m]$. The clients have different numbers of total images and different class distributions. Similar to [65], we use $M = 100/20$ for FMNIST/CIFAR-10, and sample 20%/40% clients at every round, respectively.

We train every FL algorithm for 100 (G-FL) with the generic model (GM) Table 2 provides the ablations.

To illustrate the difference between applying GMs and PMs in a P-FL setting, we also evaluate the We also consider a baseline that each client trains on their own data only, without communication. We train every FL algorithm for 100 rounds, with 5 local epochs in each round. We report the mean over five times of experiments with different random seeds. We evaluate the generic performance (G-FL) with the generic model (GM) on the standard generic test set. We evaluate the personalized performance (P-FL) with personalized models (PM) on the same set but re-weight the accuracy according to the clients’ class distributions and average the weighted accuracy across clients. This evaluation is more robust (essentially as the expectation) than assigning each client a specific test set.

**Our variants.** We mainly use Equation 6 as the $L_m^{BR}$ and report the Fed-RoD (hyper) version and its meta-learned version (see subsection 4.4). Table 2 provides the ablations.

**Baselines.** We compare to four G-FL methods: FEDAVG [69], FedProx [59], SCAFFOLD [47], and FedDyn [1]. We use their global models $\bar{w}$ for G-FL; their local models (i.e., $w_m$ in Figure 1) for P-FL evaluation. For P-FL methods, we compare to PFedME [20] and DITTO [58], which have global models available for G-FL evaluation. For methods with no global models (including MOCHA [83], FedPer [4], LG-FEDAVG [63], and FEDFOMO [103]), we averaged the final personalized models for G-FL evaluation. PER-FEDAVG [24] meta-learns a shared initialization for local models. For a fair comparison, we locally update it with the same epochs before P-FL evaluation. We also consider a baseline that each client trains on their own data only, without communication. Implementations are based on authors’ code if available.

To illustrate the difference between applying GMs and PMs in a P-FL setting, we also evaluate the P-FL performance using GMs, which is how FEDAVG has been applied to P-FL in literature.

5.1 Main results

Table 1 summarizes the results. Fed-RoD consistently outperforms all generic and personalized FL methods on both G-FL and P-FL accuracy. In terms of G-FL accuracy, advanced local training (i.e., SCAFFOLD, FedProx, and FedDyn) outperforms FEDAVG and personalized methods, and our Fed-RoD can have further gains by using balanced risk minimization (BSM). In terms of P-FL accuracy, most methods outperform the baseline of separate local training with individual data only, demonstrating the benefits of federated collaboration. Using PMs (i.e., local models $\{w_m\}$) clearly
leads to higher P-FL accuracy than using GMs (i.e., $\bar{w}$) for generic FL methods, which supports our claims and observations in Figure 1 and subsection 3.2. It is worth noting that the local models from generic FL methods are highly competitive to or even outperform personalized models from personalized FL methods. This provides generic FL methods with an add-on functionality to output personalized models by keeping the checkpoints on clients after local training. Our FED-RoD achieves the highest P-FL accuracy and we attribute this to the shared feature extractor that is learned via the robust balanced risk. We also investigate combining FED-RoD and FEDDyn [1], using the latter to optimize the generic model with BSM, which outperforms either ingredient in many cases. (See the supplementary material for other combinations.)

Analysis. To understand why FED-RoD improves G-FL, we visualize each local model $w_m$’s G-FL accuracy in Figure 5 (a). FED-RoD not only learns a better global model for G-FL, but also has a smaller variance across the local models’ generic heads (as their objectives are more aligned). We also show how $w_m$ deviates from $\bar{w}$ after training in Figure 5 (b). FED-RoD has a smaller variance.

Pushing the limit with Meta-BSM (cf. subsection 4.4). We sample 10 images for each class (only 0.2% of the overall training set) from the training set as the meta set. We compare to [106] that concatenates the meta set to clients’ local data. The results in Table 1 are encouraging. With a very small meta set, FED-RoD outperforms [106] by 1% to 14% on accuracy across different settings, validating the importance of balanced losses and how to set them up dynamically via meta-learning.

FED-RoD (hyper) for future clients. To validate the generalizability to new clients, we build on the CIFAR-10 (Dir(0.3)) setting but split the training data into 100 clients (50 are in training; 50 are new) and train for 100 rounds (sampling 40% of the 50 training clients every round). After that, we evaluate on the 50 new clients individually, either using the global model directly or fine-tuning it with clients’ data for several local steps. Figure 5 (c) shows the averaged accuracy. Without any further local training (i.e., fine-tuning), FED-RoD (hyper) can already generate personalized models, and outperforms others methods stably with fine-tuning.

5.2 Further analysis (more in the supplementary materials, including deeper networks)

Ablation study. We compare several variants of FED-RoD (cf. Figure 4), with one head (reduced to FEDAVG) or different networks (linear/hyper). We use CIFAR-10 with Dir(0.3)-non-IID clients. As shown in Table 2, FEDAVG with BSM significantly improves G-FL but degrades in P-FL. FED-RoD remedies it by training a decoupled personalized head. We note that, FED-RoD does not merely fine-tune the global model with clients’ data (cf. subsection 4.2). We compare different balanced losses in Table 3. Using more advanced losses [10, 77, 97] is better than simple re-weighting [37].

Class-imbalanced global distributions. FED-RoD is robust even when the global distribution is class-imbalanced since BRM seeks to learn every class well. See the supplementary material.
Comparisons to P-FL methods. While G-FL algorithms like FedAvg can already achieve competitive P-FL accuracy compared to P-FL algorithms, there are certainly several scenarios where P-FL algorithms might be preferred, *e.g.*, robustness against adversaries [58] (see the supplementary material). We suggest that future work on P-FL should report results in these challenging scenarios.

6 Conclusion

The community of federated learning (FL) has been dedicated to either learning a better generic model or personalized models. We show that these two contrasting goals can be achieved simultaneously via a novel two-loss, two-predictor FL framework Fed-ROD, which is designed to excel in both tasks.

Acknowledgments

This research is supported by the OSU GI Development funds. We are thankful for the generous support of computational resources by Ohio Supercomputer Center and AWS Cloud Credits for Research.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *ICLR*, 2021. 2, 3, 5, 6, 8, 9, 20, 21, 25, 26

[2] Alekh Agarwal, John Langford, and Chen-Yu Wei. Federated residual learning. *arXiv preprint arXiv:2003.12880*, 2020. 3

[3] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhandhur. Invariant risk minimization games. In *ICML*, 2020. 3, 19

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00873*, 2019. 3, 8

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 3, 19

[6] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010. 5

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzucchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. 3

[8] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. 5

[9] Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019. 3

[10] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arelchiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Conference on Neural Information Processing Systems*, 2019. 2, 6, 9, 17, 18, 20, 24

[11] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, 2016. 21

[12] Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *ICCV*, 2017.

[13] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Classifier and exemplar synthesis for zero-shot learning. *IJCV*, 128(1):166–201, 2020. 21

[14] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and X. He. Federated meta-learning with fast convergence and efficient communication. *arXiv: Learning*, 2018. 3
[15] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. In *ICLR*, 2021. 3

[16] Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019. 3, 4

[17] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 2, 17

[18] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020. 3

[19] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Distributionally robust federated averaging. *NeurIPS*, 33, 2020. 3

[20] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *NeurIPS*, 2020. 1, 2, 3, 4, 8, 21, 26

[21] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):59–71, 2020. 3

[22] An Evgeniou and Massimiliano Pontil. Multi-task feature learning. In *NeurIPS*, 2007. 3

[23] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *KDD*, 2004. 3

[24] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *NeurIPS*, 2020. 3, 8, 21, 25, 26, 27

[25] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 5

[26] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015. 3, 5

[27] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE, 2012. 5

[28] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 20

[29] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *ICLR*, 2017. 2, 3, 7, 18, 21

[30] Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019. 1, 3

[31] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020. 3, 4

[32] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. In *NeurIPS*, 2020. 1, 2, 3, 4

[33] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. In *NeurIPS*, 2020. 3

[34] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. 2, 3

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 27

[36] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019. 3, 5, 8, 21

[37] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *ECCV*, 2020. 3, 5, 6, 9, 18
[38] Yutao Huang, Lingyang Chu, Z. Zhou, Lanjun Wang, J. Liu, Jian Pei, and Yanxin Zhang. Personalized cross-silo federated learning on non-iid data. In AAAI, 2021. 3

[39] Laurent Jacob, Francis Bach, and Jean-Philippe Vert. Clustered multi-task learning: A convex formulation. In NeurIPS, 2009. 3

[40] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In Proc. of the Int’l Conf. on Artificial Intelligence, 2000. 5

[41] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. Intelligent data analysis, 6(5):429–449, 2002. 5

[42] Yihan Jiang, Jakub Konečný, Keith Rush, and S. Kannan. Improving federated learning personalization via model agnostic meta learning. ArXiv, abs/1909.12488, 2019. 3

[43] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. Journal of Big Data, 6(1):27, 2019. 5

[44] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977, 2019. 1

[45] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In ICLR, 2020. 6, 17

[46] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. arXiv preprint arXiv:2008.03606, 2020. 3, 5, 20

[47] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In ICML, 2020. 2, 3, 6, 8, 20, 25, 26

[48] A Khaled, K Mishchenko, and P Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In AISTATS. 2020. 3

[49] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE transactions on neural networks and learning systems, 29(8):3573–3587, 2017. 6, 17

[50] M. Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In NeurIPS, 2019. 3

[51] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492, 2016. 3

[52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 7

[53] V. Kulkarni, Milind Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pages 794–797, 2020. 3

[54] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. TPAMI, 36(3):453–465, 2013. 21

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. 7, 8, 21, 27

[56] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581, 2019. 3, 4

[57] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smithy. Feddane: A federated newton-type method. In 2019 53rd Asilomar Conference on Signals, Systems, and Computers, 2019. 3

[58] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through. arXiv preprint arXiv:2012.04221, 2020. 3, 4, 5, 8, 10, 22, 25, 26, 27
[59] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In MLSys, 2020. 2, 3, 5, 6, 8, 20, 21, 25, 26

[60] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In ICLR, 2020. 1, 3, 5, 21

[61] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fed[bn]: Federated learning on non-[iid] features via local batch normalization. In ICLR, 2021. 3

[62] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In CVPR, 2020. 20

[63] Paul Pu Liang, Terrance Liu, Liu Ziyin, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. arXiv preprint arXiv:2001.01523, 2020. 3, 8, 26

[64] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local sgd with lower communication complexity. arXiv preprint arXiv:1912.12844, 2019. 3

[65] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In NeurIPS, 2020. 3, 8, 21

[66] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In Conference on Neural Information Processing Systems, 2018. 5

[67] Grigory Malinovskiy, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtarik. From local sgd to local fixed-point methods for federated learning. In ICML, 2020. 3, 20

[68] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619, 2020. 3

[69] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In AISTATS, 2017. 1, 2, 3, 4, 8, 16, 17, 21, 25, 26

[70] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In ICML, 2019. 3

[71] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In ICML, 2013. 3, 5

[72] Reese Pathak and Martin J Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. In NeurIPS, 2020. 3, 20

[73] Daniel Peterson, Pallika Kanani, and Virendra J Marathe. Private federated learning with domain adaptation. arXiv preprint arXiv:1912.06733, 2019. 3

[74] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federative optimization. In ICLR, 2021. 3

[75] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. arXiv preprint arXiv:1909.13014, 2019. 3

[76] Matthais Reissser, Christos Louizos, Efstratios Gavves, and Max Welling. Federated mixture of experts, 2021. URL https://openreview.net/forum?id=fgrdmztE4Y. 3

[77] Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Balanced meta-softmax for long-tailed visual recognition. In NeurIPS, 2020. 2, 6, 9, 17, 18, 19, 20

[78] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In ICML, 2018. 7, 19

[79] Sebastien Ruder. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098, 2017. 3

[80] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In ICML, 2021. 3

[81] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In NeurIPS, 2019. 7, 19
[105] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. 2010. 3

[106] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018. 1, 3, 5, 7, 8, 9, 19, 20, 26

[107] Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. arXiv preprint arXiv:1708.01012, 2017. 1

[108] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. arXiv preprint arXiv:2009.07999, 2020. 3

[109] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In NeurIPS, 2010. 1
Supplementary Material

We provide details omitted in the main paper.

- Appendix A: additional details of FED-RoD (cf. section 3 and section 4 of the main paper).
- Appendix B: additional comparison to related work (cf. section 2 and section 3 of the main paper).
- Appendix C: details of experimental setups (cf. section 5 of the main paper).
- Appendix D: additional experimental results and analysis (cf. section 3 and section 5 of the main paper).

A Additional Details of FED-RoD

A.1 Additional background (cf. subsection 3.1 of the main paper)

In the generic federated learning (FL) setting, the goal is to construct a single “global” model that can perform well for test data from all the clients. Let \( w \) denote the parameters of the model, for a classification problem whose label space is \( \mathcal{C} \), a commonly used loss is the cross entropy,

\[
\ell(x, y; w) = -\log \frac{\exp(g_c(x; w))}{\sum_{c \in \mathcal{C}} \exp(g_c(x; w))},
\]  

(12)

where \( g_c(x; w) \) is the model’s output logit for class \( c \).

We note that, the concepts of global vs. local models and generic vs. personalized models should not be confused. No matter which task (generic or personalized) an FL algorithm focuses on, as long as it has the local training step, it generates local models; as long as it has the global aggregation step (of the entire model), it generates a global model. For instance, FEDAVG [69] aims for generic FL but it creates both the global and local models.

A.2 Overview of FED-RoD

For generic predictions, FED-RoD performs feature extraction \( z = f(x; \theta) \), followed by \( h^G(z; \psi) \).

For personalized predictions, FED-RoD performs \( z = f(x; \theta) \), followed by \( h^G(z; \psi) + h^P(z; \phi_m) \).

The element-wise addition is performed at the logit level. That is, \( g_c(x; w) \) in Equation 12 can be re-written as

\[
ge_c(x; \{\theta, \psi, \phi_m\}) = \begin{cases} 
  h^G_c(z; \psi) & \text{Generic model,} \\
  h^G_c(z; \psi) + h^P_c(z; \phi_m) & \text{Personalized model,}
\end{cases}
\]

(13)

where \( z = f(x; \theta) \) is the extracted feature.

The overall training process of FED-RoD iterates between the local training and global aggregation steps. In local training, FED-RoD aims to minimize the following objective

\[
\mathcal{L}^{BR}_m(\{\theta, \psi\}) + \mathcal{L}_m(\{\theta, \psi, \phi_m\}).
\]

(14)

The empirical risk \(\mathcal{L}_m(w_m = \{\theta, \psi, \phi_m\})\) is defined as \(\sum_{m=1}^M \frac{1}{|D_m|} \sum_{i \in D_m} \ell(x_i, y_i; w_m)\), where \( D_m = \{(x_i, y_i)\}_{i=1}^{|D_m|} \) is the training data of client \( m \). We will introduce more options of the balanced risk \( \mathcal{L}^{BR}_m \) in subsection A.3. We optimize Equation 14 via stochastic gradient descent (SGD). We updates \( \theta, \psi, \) and \( \phi_m \) in a single forward-backward pass, which consumes almost the same computation cost as FEDAVG. For \(\mathcal{L}_m(\{\theta, \psi, \phi_m\})\), we do not derive gradients w.r.t. \( \theta \) and \( \psi \).

We emphasize that, according to subsection 4.2 of the main paper, the finally learned parameters of FED-RoD (linear) are \( \theta, \psi, \) and \( \{\phi_m\}_{m=1}^M \). We then plug them into Equation 13 for predictions.
In algorithm 1 and algorithm 2, we provide pseudocode of our FED-ROD algorithm.

**Algorithm 1: FED-ROD (linear) (Federated Robust Decoupling)**

**Server input**: initial global model parameter $\theta$ and $\psi$;

**Client $m$’s input**: initial local model parameter $\phi_m^*$, local step size $\eta$, local labeled data $D_m$;

for $r \leftarrow 1$ to $R$ do

\begin{itemize}
  \item Sample clients $S \subseteq \{1, \cdots, N\}$;
  \item Communicate $\theta$ and $\psi$ to all clients $m \in S$;
  \item for each client $m \in S$ in parallel do
    \begin{itemize}
      \item Initialize $\theta \leftarrow \bar{\theta}$, $\psi \leftarrow \bar{\psi}$, and $\phi_m \leftarrow \phi_m^*$;
      \item $\{\theta_m^*, \psi_m^*, \phi_m^*\} \leftarrow \text{Client local training}(\{\theta, \psi, \phi_m\}, D_m, \eta)$; \hspace{1cm} [Equation 8 and Equation 9]
    \end{itemize}
  \item Communicate $\theta_m^*$ and $\psi_m^*$ to the server;
  \item Construct $\bar{\theta} = \sum_{m \in S} \frac{|D_m|}{\sum_{m \in S} |D_m|} \theta_m^*$;
  \item Construct $\bar{\psi} = \sum_{m \in S} \frac{|D_m|}{\sum_{m \in S} |D_m|} \psi_m^*$;
\end{itemize}

\textbf{Server output}: $\bar{\theta}$ and $\bar{\psi}$;

\textbf{Client $m$’s output}: $\phi_m^*$.

**Algorithm 2: FED-ROD (hyper) (Federated Robust Decoupling)**

**Server input**: initial global model parameter $\bar{\theta}$, $\bar{\psi}$, and $\bar{\nu}$;

**Client $m$’s input**: local step size $\eta$, local labeled data $D_m$;

for $r \leftarrow 1$ to $R$ do

\begin{itemize}
  \item Sample clients $S \subseteq \{1, \cdots, N\}$;
  \item Communicate $\bar{\theta}$, $\bar{\psi}$, and $\bar{\nu}$ to all clients $m \in S$;
  \item for each client $m \in S$ in parallel do
    \begin{itemize}
      \item Initialize $\theta \leftarrow \bar{\theta}$, $\psi \leftarrow \bar{\psi}$, and $\nu \leftarrow \bar{\nu}$;
      \item $\{\theta_m^*, \psi_m^*, \nu_m^*\} \leftarrow \text{Client local training}(\{\theta, \psi, \nu\}, D_m, \eta)$; \hspace{1cm} [Equation 11]
      \item Communicate $\theta_m^*$, $\psi_m^*$, and $\nu_m^*$ to the server;
    \end{itemize}
  \item Construct $\bar{\theta} = \sum_{m \in S} \frac{|D_m|}{\sum_{m \in S} |D_m|} \theta_m^*$;
  \item Construct $\bar{\psi} = \sum_{m \in S} \frac{|D_m|}{\sum_{m \in S} |D_m|} \psi_m^*$;
  \item Construct $\bar{\nu} = \sum_{m \in S} \frac{|D_m|}{\sum_{m \in S} |D_m|} \nu_m^*$;
\end{itemize}

\textbf{Server output}: $\bar{\theta}$, $\bar{\psi}$, and $\bar{\nu}$ (for personalized model generation).

### A.3 Balanced risk minimization (BRM)

To learn a generic model, standard federated learning (e.g., FEDAVG [69]) aims to optimize $L(w)$ in Equation 1 of the main paper. In theory, the overall objective $L(w)$ is equal to the expected objective $\mathbb{E}[L_m(w)]$ for client $m$, if client $m$’s data $D_m$ are IID partitioned from $D$. Here, the expectation is over different $D_m$ partitioned from $D$. In reality, $L_m$ could diverge from $L$ due to non-IID partitions of the aggregated data $D$ into clients’ data. That is, $\mathbb{E}[L_m(w)] \neq \mathbb{E}[L_m'(w)] \neq L(w)$. We mitigate the non-IID situation by directly adjusting $L_m$ such that $\mathbb{E}[L_m(w)] \approx \mathbb{E}[L_m'(w)] \approx L(w)$.

Essentially, $L_m(w)$ is the client’s empirical risk, which could be different among clients if their class distribution $P_m(y)$ are different. We, therefore, propose to turn the empirical risk $L_m$ into a class-balanced risk $L_m^{BR}$ by replacing $L$ in Equation 12 with a class-balanced loss [10, 17, 45, 49, 77, 97]. The class-balanced loss attempts to make the learned model robust to different training class distributions, such that the learned model can perform well for all the test classes. In other words, the class-balanced loss is designed with an implicit assumption that the test data will be class-balanced, even though the training data may not be. Table 4 summarizes some popular class-balanced losses.
We implement the meta-model $H_c$ which records the proportion of class $a$.

To this end, instead of learning a specific prediction head $\phi$ for each client, we propose to learn a meta-model $H^P(a_m; \nu)$ with a shared meta-parameter $\nu$. The input to $H^P$ is a vector $a_m \in \mathbb{R}^{||C||}$, which records the proportion of class $c \in C$ in client $m$’s data. The output of $H^P$ is $\Phi_m$ for $h^P$. In other words, $H^P$ can adaptively output personalized prediction heads for clients given their local class distributions $a_m$.

We implement the meta-model $H^P$ by a hypernetwork [29], which can be seen as a lightweight classifier generator given $a_m$. This lightweight hypernetwork not only enables clients to collaboratively learn a module that can generate customized models, but also allows any (future) clients to immediately generate their own personalized predictors given their local class distribution $a_m$ as input, even without training. We construct the hypernetwork by two fully-connected (FC) layers.
A.5 Extension with meta-learning for the improved BSM loss

Fed-RoD incorporates a balanced loss to learn the generic model. Here we study a more advanced way to derive such balanced loss with meta-learning. Inspired by [78, 81] and the FL scenario proposed by [106], we seek to combine the BSM loss and re-weighting as $\sum q_{m,y} \ell_{BSM}(x, y; w)$, where $q_{m,y}$ is meta-learned with a small balanced meta dataset $\mathcal{D}_\text{meta}$ provided by the server. (See Table 4 for a comparison.) The $\mathcal{D}_\text{meta}$ should have a similar distribution to the future test data. We implement this idea with the Meta-Weight Net (MWNet) [81] with learnable parameter $\bar{\phi}$.

In addition, we notice that the original BSM loss $\ell_{BSM} = -\log \frac{N_{\gamma_m} \exp(g_m(x, w))}{\sum_{x \in N_{\gamma_m}} \exp(g_m(x, w))}$ has a hyper-parameter $\gamma$ which is set to be 1 via validation [77]. However, in federated learning it can be hard to tune such a hyperparameter due to the large number of non-IID clients. Therefore, we propose to learn a client-specific $\gamma_m$ with meta-learning for $\ell_{BSM}$. More specifically, given a meta-learning rate $\eta$, the meta-learning process involves the following iterative steps:

1. Compute the Meta-BSM loss with a mini-batch $B \sim \mathcal{D}_m$: i.e., $\forall (x, y) \in B$, compute $\ell_{BSM}(x, y; w_m)$.
2. Predict the example weights with $q_{m,y} = \text{MWNet}(\ell_{BSM}(x, y; w_m); \bar{\phi}_m), \forall (x, y) \in B$.
3. Re-weight the Meta-BSM loss: $L_{BR}^{\text{meta}}(w_m) = \sum (x,y) \in B q_{m,y} \ell_{BSM}(x, y; w_m)$, and perform one step of gradient descent to create a duplicated model $\tilde{w}_m = w_m - \eta \nabla_w L_{BR}^{\text{meta}}$.
4. Computes the loss on the meta dataset $\mathcal{D}_\text{meta}$ using the duplicated model: $L_{BR}^{\text{meta}}(\tilde{w}_m) = \sum (x,y) \in \mathcal{D}_\text{meta} q_{m,y} \ell_{BSM}(x, y; \tilde{w}_m)$, followed by updating $\gamma_m \leftarrow \gamma_m - \eta \nabla_{\gamma_m} L_{BR}^{\text{meta}}$ and $\bar{\phi}_m \leftarrow \bar{\phi}_m - \eta \nabla_{\bar{\phi}_m} L_{BR}^{\text{meta}}$.
5. Update the model: $w_m \leftarrow w_m - \eta \nabla_w L_{BR}^{\text{meta}}(w_m)$.

Throughout the federated learning process, $\gamma_m$ and $q_{m,y}$ are dynamically learned with meta-learning for different clients and rounds.

A.6 Connection to Invariant Risk Minimization Games (IRMG)

Fed-RoD is inspired by a recently proposed machine learning framework Invariant Risk Minimization (IRM) [5] and its extension Invariant Risk Minimization Games (IRMG) [3].

Suppose that the whole dataset is collected from many environments, where data from each environment is associated with its characteristic. IRM introduces the concept of learning an invariant predictor. (Note that, in IRM the learner can access data from all the environments; thus, it is not for an FL setting.) Given the training data partition, IRM aims to learn an invariant feature extractor $z = f(x; \theta)$ and a classifier $h(z; \psi)$ that achieves the minimum risk for all the environments.

The concept of environments can be connected to clients’ private local data in FL which are often non-IID. That is, given $M$ environments, we can re-write IRM in a similar expression to Equation 7 in the main paper

$$\min_{\theta, \psi} L^{\text{IRM}}(\theta, \psi) = \sum_{m=1}^{M} L_m(\theta, \psi),$$

s.t. $\psi \in \arg \min_{\psi'} L_m(\theta, \psi'), \forall m \in [M].$  \hspace{1cm} (15)

Unfortunately, IRM is intractable to solve in practice given the constraint that every environment relies on the same parameters [3]. IRMG relaxes it by reformulating the classifier $\psi$ as an ensemble of environment-specific classifiers (by averaging over model weights) $\bar{\psi} = \frac{1}{M} \sum_m \phi_m$.

$$\min_{\theta, \phi} L^{\text{IRMG}}(\theta, \bar{\phi}) = \sum_{m=1}^{M} L_m(\theta, \phi),$$

s.t. $\phi_m \in \arg \min_{\phi_{m'}} L_m(\theta, \phi_{m'}), \forall m \in [M].$  \hspace{1cm} (16)

$$\min_{\theta, \phi} L^{\text{IRMG}}(\theta, \bar{\phi}) = \sum_{m=1}^{M} L_m(\theta, \phi),$$

s.t. $\phi_m \in \arg \min_{\phi_{m'}=m} L_m(\theta, \phi_{m'}), \forall m \in [M].$
IRMG is proved to optimize the same invariant predictor of IRM when it converges to the equilibrium in game theory, and it holds for a large class of non-linear classifiers. IRMG is solved through iterative optimization: (1) training the feature extractor $\theta$ with centralized data (i.e., aggregated data from all environments), (2) training the environment-specific classifiers $\phi_m$ on the data of each environment $D_m$, and (3) updating the main classifier through weight averaging $\phi = \frac{1}{M} \sum_m \phi_m$.

We highlight the similarity between IRMG and FedRoD: both are training a strong generic feature extractor and a set of personalized classifiers. For predictions on data of client (environment) $m$ in Equation 18, IRMG uses $\hat{y} = \frac{1}{M}(\phi_m^\top z + \sum_{m' \neq m} \phi_{m'}^\top z)$; FedRoD’s personalized model is $\hat{y} = h^G(z; \psi) + h^P(z; \phi_m)$. We can connect IRMG to FedRoD by re-writing its prediction as $h^G(z; \phi) := \phi^\top z = \frac{1}{M} \sum_m \phi_m^\top z$ and $h^P(z; \phi_m) := \frac{1}{M} (\phi_m^\top z - \phi_m'\top z)$, where $\phi_m'$ is the client $m$’s model in the previous round/iteration of learning.

IRMG can not be applied directly to federated learning for the following reasons. First, centralized training of the feature extractor is intractable since clients’ data are not allowed to be aggregated to the server. Second, to perform the iterative optimization of IRMG, the clients are required to communicate every step, which is not feasible in FL due to communication constraints.

## B Comparison to Related Work

### B.1 Systematic overhead

FedRoD has similar computation cost, communication size, and number of parameters as FedAvg. We discuss the difference between FedRoD and existing generic FL methods from a system view. FedProx [59] proposes a proximal term to prevent client from diverging from the server model, which is more robust to the heterogeneous system. SCAFFOLD [47] imposes a gradient correction during client training. Maintaining such a correction term, however, doubles the size of communication. FedDyn [1] resolves the communication cost issue by introducing a novel dynamic regularization. However, it requires all users to maintain their previous models locally throughout the FL process, which is not desired when users have memory and synchronization constraints.

### B.2 Follow-up works of FedAvg

Several recent works [46, 47, 106] have shown that, with multiple steps of local SGD updates, the local model $w_m$ would drift away from each other, leading to a degenerated global model $\bar{w}$ that deviates from the solution of Equation 1 of the main text.

One way to mitigate this is to modify the local training objective (cf. Equation 2 of the main text). For instance, FedProx [59] introduced a regularizer with respect to $\bar{w}$,

$$\min_w L_m(w) + \frac{\lambda}{2} \|w - \bar{w}\|^2.$$  

FedDyn [1] further added a dynamic term based on the local model of the previous round $w_{m}'$,

$$\min_w L_m(w) + \langle \nabla L_m(w_{m}'), w \rangle + \frac{\lambda}{2} \|w - \bar{w}\|^2.$$  

These regularizers aim to stabilize local training and align the objectives among clients. Some other works did not change the objectives but introduced control variates or momentum to correct the local gradient [46, 47], or designed a new optimizer more suitable for decentralized learning [67, 72, 99].

It is worth mentioning, in most of these works, the empirical risk $L_m(w)$ still plays an important role in driving the local model update. Since $L_m(w)$ directly reflects the (non-IID) client data distribution, the learned local models are indeed strong candidates for personalized models.

### B.3 Class-imbalanced learning

Class-imbalanced learning attracts increasing attention for two reasons. First, models trained under this scenario using empirical risk minimization perform poorly on minor classes of scarce training data. Second, many real-world data sets are class-imbalanced by nature [28, 87, 88]. In this paper, we employ a mainstream approach, cost-sensitive learning [10, 62, 77, 85, 97], which adjusts the training objective to reflect class imbalance so as to train a model that is less biased toward major classes.
B.4 Zero-shot learning

Our design choice of parameterizing the personalized prediction head with clients’ class distributions is reminiscent of zero-shot learning [11–13, 54, 93], whose goal is to build an object classifier based on its semantic representation. The key difference is that we build an entire fully-connected layer for FL, not just a single class vector. We employ hypernetworks [29] for efficient parameterization.

C Implementation Details

Implementation. We adopt ConvNet [55] following the existing works [1, 69, 86]. For FMNIST, it contains 2 Conv layers and 2 FC layers. The Conv layers have 32 and 64 channels, respectively. The FC layers are with 50 neurons as the hidden size and 10 neurons for 10 classes as outputs, respectively. For CIFAR-10, it contains 3 Conv layers and 2 FC layers. The Conv layers have 32, 64, and 64 channels, respectively. The FC layers are with 64 neurons as the hidden size and 10 neurons for 10 classes as outputs, respectively. To implement hypernetworks in Fed-RoD, we use a simple 2-FC ReLU network with hidden size 16 for FMNIST and 32 for CIFAR-10.

We use standard pre-processing, where both FMNIST and CIFAR-10 images are normalized. FMNIST is trained without augmentation. The 32 × 32 CIFAR-10 images are padded 2 pixels each side, randomly flipped horizontally, and then randomly cropped back to 32 × 32.

We train every method for 100 rounds. We initialize the model weights from normal distributions. As mentioned in [60], the local learning rate must decay along the communication rounds. We initialize it with 0.01 and decay it by 0.99 every round, similar to [1]. Throughout the experiments, we use the SGD optimizer with weight decay 1e−5 and a 0.9 momentum. The mini-batch size is 40. In each round, clients perform local training for 5 epochs. We report the mean over five times of experiments with different random seeds.

For FedProx [59], the strength of regularization λ is selected from [1e−2, 1e−3, 1e−4]. For FedDyn [1], the strength of regularization λ is selected from [1e−1, 1e−2, 1e−3] as suggested in [1]. For PER-FedAvg [24], the meta-learning rate β is selected from [1e−2, 1e−3, 1e−4]. For PFedME [20], the strength of regularization λ is selected from [15, 20, 30]. Fed-RoD introduces no extra hyperparameters on top of FedAvg.

For the generic and personalized heads of Fed-RoD, we study using 1 ~ 4 FC layers but do not see a notable gain by using more layers. We attribute this to the well-learned generic features. Thus, for all our experiments on Fed-RoD, we use a single FC layer for each head.

We run our experiments on four GeForce RTX 2080 Ti GPUs with Intel i9-9960X CPUs.

Evaluation. Both datasets and the non-IID Dirichlet simulation are widely studied and used in literature [1, 36, 65]. We use the standard balanced test set D_test for evaluation on generic FL (G-FL):

\[
\text{G-FL accuracy} : \frac{1}{|D_{test}|} \sum_i 1(y_i = \hat{y}_{i,G}),
\]

where \( \hat{y}_{i,G} \) here is the predicted label (i.e., \( \arg\max \) over the logits). For evaluation on personalized FL (P-FL), we still apply \( D_{test} \) but weight instances w.r.t. each client’s class distribution:

\[
\text{P-FL accuracy} : \frac{1}{\bar{M}} \sum_m \frac{\sum_i p_m(y_i) 1(y_i = \hat{y}_{i,P})}{\sum_i p_m(y_i)}.
\]

We do so instead of separating \( D_{test} \) into separate clients’ test sets in order to avoid the variance caused by how we split test data. What we compute is essentially the expectation over the splits. We have verified that the difference of the two evaluation methods is negligible.

In Table 1 of the main paper and some other tables in the supplementary material, we evaluate G-FL by an FL algorithm’s generic (usually the global) model, denoted as GM. We evaluate P-FL by an FL algorithm’s personalized models (or local models of a G-FL algorithm), denoted as PM. For P-FL, we also report the generic model’s accuracy following the literature to demonstrate the difference.
D Additional Experiments and Analyses

Here we provide additional experiments and analyses omitted in the main paper. We validate our claims in the main paper and the designs of our proposed FED-ROD via the following experiments:

- subsection D.1: personalized models emerge from local training of generic federated learning (cf. subsection 3.2, subsection 3.3, and subsection 5.1 in the main paper).
- subsection D.2: balanced risk minimization (BRM) improves generic-FL performance (cf. subsection 5.1 and subsection 5.2 in the main paper).
- subsection D.3: the roles of FED-ROD’s generic and personalized heads (cf. subsection 4.2 and subsection 5.2 in the main paper).
- subsection D.4: personalization with hypernetworks (cf. subsection 5.1 in the main paper).
- subsection D.5: robustness to class-imbalanced global data.
- subsection D.6: compatibility of FED-ROD with other G-FL algorithms (cf. subsection 5.1 in the main paper).
- subsection D.7: comparison to personalized FL algorithms (cf. subsection 5.2 in the main paper).
- subsection D.8: ablation studies and discussions on FED-ROD (cf. subsection 5.2 in the main paper).

D.1 Personalized models emerge from local training of generic federated learning

As mentioned in section 3 in the main paper, personalized FL algorithms usually impose an extra regularizer (cf. Equation 3 and Equation 4 of the main paper) during local training, but do not re-initialize the local models by the global models at every round. In contrast, generic FL algorithms like FEDAVG do not impose extra regularization but re-initialize the local models at every round. Here in Figure 6, we monitor the two loss terms, \( \sum_{m} |D_m| L_m(w_m) \) and \( \sum_{m} |D_m| \|w_m - \bar{w}\|^2 \) (cf. Equation 3 and Equation 4 of the main paper), for FEDAVG and a state-of-the-art personalized FL algorithm Ditto [58] at the end of each local training round. (Ditto does include the \( L_2 \) regularizer in training the personalized models.) Ditto achieves a lower empirical risk (i.e., the first term), likely due to the fact that it does not perform re-initialization.Surprisingly, FEDAVG achieves a much smaller regularization term (i.e., the second term) than Ditto, even if it does not impose such a regularizer in training. We attribute this to the strong effect of regularization by re-initialization: as mentioned in subsection 3.3 of the main paper, re-initialization is equivalent to setting the regularization coefficient \( \lambda \) as infinity. We note that, the reason that the regularization term of Ditto increases along the communication rounds is because ever time the global model \( \bar{w} \) is updated, it moves sharply away from the local model \( w_m \). Thus, even if the regularization term is added into local training, it cannot...
be effectively optimized. This analysis suggests that the local models of generic FL algorithms are more regularized than the personalized models of personalized FL algorithms. The local models of generic FL algorithms are thus strong candidates to be evaluated in the personalized FL setting.

D.2 Balanced risk minimization (BRM) improves generic-FL performance

To understand why Fed-RoD outperforms other generic methods in the G-FL accuracy, we visualize each local model $w_m$’s G-FL accuracy after local training in Figure 7 (both datasets with Dir(0.3)). Methods rely on ERM suffer as their local models tend to diverge. Figure 8 further shows that the variances of local weight update $\Delta w_m = w_m - \bar{w}$ across clients are smaller for Fed-RoD, which result from a more consistent local training objective.

In Figure 9, we further compare the G-FL accuracy among FedAvg, Fed-RoD with the original BSM loss, and Fed-RoD with the Meta-BSM loss introduced in subsection A.5 along the training process (i.e., training curve). The local models of FedAvg tend to diverge from each other due to the non-IID issue, resulting in high variances and low accuracy of G-FL. The global aggregation does improve the G-FL accuracy, validating its importance in federated learning. The local training in Fed-RoD (BSM) not only leads to a better global model, but also has smaller variances and higher accuracy for the local models (as their objectives are more aligned). With the help of meta dataset and meta-learning, Fed-RoD (Meta-BSM) yields even better G-FL performance for both global models and local models, and has much smaller variances among local models’ performance, demonstrating the superiority of using meta-learning to learn a balanced objective.

D.3 The roles of Fed-RoD’s generic and personalized heads

To demonstrate that Fed-RoD’s two heads learn something different, we plot in Figure 10 every local model’s generic prediction and personalized prediction on its and other clients’ data (i.e., P-FL accuracy). The generic head performs well in general for every client’s test data. The personalized head could further improve for its own data (diagonal), but degrade for others’ data.
D.4 Personalization with hypernetworks

**Fed-RoD (hyper)** learns the personalized head with hypernetworks as introduced in subsection A.4. The goal is to learn a hypernetwork such that it can directly generate a personalization prediction head given client’s class distribution, without further local training. **Figure 11** shows the training (convergence) curves on CIFAR-10 Dir(0.3). The hypernetwork (globally aggregated, before further local training) can converge to be on par with that after local training. In the main paper (cf. Figure 5 (c)), we also show that it servers as a strong starting point for future clients — it can generate personalized models simply with future clients’ class distributions. That is, the clients may not have labeled data, but provide the hypernetwork with their preference/prior knowledge. It can also be used as the warm-start model for further local training when labeled data are available at the clients.

Table 6 provides the P-FL results for the new 50 clients studied in subsection 5.1 and Figure 5 (c) in the main paper. Except for **Fed-RoD (hyper)**, the accuracy before local training is obtained by the global model. The best personalized model after local training is selected for each client using a validation set. **Fed-RoD (hyper)** notably outperforms other methods before or after local training.

D.5 Class-imbalanced global distributions

In the real world, data frequency naturally follows a long-tailed distribution, rather than a class-balanced one. Since the server has no knowledge and control about the whole collection of the clients’ data, the clients data may collectively be class-imbalanced. This adds an additional challenge for the server to learn a fair and class-balanced model. We follow the setup in [10] to transform FMNIST and CIFAR-10 training sets into class-imbalanced versions, in which the sample sizes per class follow an exponential decay. The imbalanced ratio (IM) is controlled as the ratio between sample sizes of
Figure 11: P-FL accuracy of hypernetwork before and after local training.

Table 6: CIFAR-10 Dir(0.3) P-FL accuracy on future non-IID clients.

| Method / Local Training | Before | Best |
|-------------------------|--------|------|
| FEDAVG [59]             | 56.2   | 76.0 |
| FEDPROX [59]            | 56.9   | 75.9 |
| SCAFFOLD [47]           | 57.1   | 75.4 |
| FEDDYN [1]              | 61.7   | 76.2 |
| PER-FEDAVG [24]         | 60.0   | 79.8 |
| FED-RoD (linear)        | 62.4   | 80.2 |
| FED-RoD (hyper)         | 75.7   | 81.5 |
| FED-RoD (hyper) +FEDDYN | 77.1   | 83.5 |

the most frequent and least frequent classes. Here we consider IM = 10 and IM = 100. The generic test set remains class-balanced.

Table 7 shows that FED-RoD remains robust on both generic accuracy and client accuracy consistently. We see that FEDDYN also performs well, especially on FMNIST of which the setup has more clients (100) but a lower participation rate (20%). By combining FEDDYN with FED-RoD, we achieve further improvements.

Essentially, the generic FL methods (except for FED-RoD) are optimizing toward the overall class-imbalanced distribution rather than the class-balanced distribution. In Table 8, we further examine the G-FL accuracy on a class-imbalanced test set whose class distribution is the same as the global training set. FED-RoD still outperforms other methods, demonstrating that FED-RoD learns a robust, generic, and strong model.

D.6 Compatibility of FED-RoD with other G-FL algorithms

As mentioned in the main paper, other G-FL algorithms like FEDDYN [1] can be incorporated into FED-RoD to optimize the generic model (using the balanced risk). We show the results in Table 9, following Table 1 of the main paper. Combining FED-RoD with SCAFFOLD [47], FEDDYN [1], and FEDPROX [59] can lead to higher accuracy than each individual algorithm along in several cases.

D.7 Comparison to personalized FL algorithms

From Table 1 in the main paper and Table 7, the personalized FL algorithms are usually outperformed by local models of generic FL algorithms in terms of the P-FL accuracy (i.e., the PM column). The gap is larger when client data are more IID, especially for P-FL methods whose personalized models do not explicitly rely on weight averaging of other clients’ models (e.g., MOCHA, LG-FEDAVG, and PFEDME). Some P-FL methods can not even outperform local training alone. A similar observation is also reported in FEDFOMO [103]. These observations justify the benefits of FL that similar clients can improve each other by aggregating a global model and updating it locally, while the benefits might decay for very dissimilar clients.

To further demonstrate the effect of building a global model and re-initializing the local/personalized models using it (cf. section 3 in the main paper), we investigate Ditto [58], a state-of-the-art
Table 7: Class-imbalanced global training distribution. *: methods with no global models and we combine their P-FL models. Gray rows: meta-learning with 100 labeled server data.

| Dataset | Non-IID / Imbalance Ratio | Dir(0.3), IM10 | Dir(0.3), IM100 | Dir(0.6), IM10 | Dir(0.6), IM100 |
|---------|--------------------------|---------------|----------------|---------------|----------------|
| Test Set | G-FL | P-FL | G-FL | P-FL | G-FL | P-FL | G-FL | P-FL |
| Method / Model | GM | GM | PM | GM | GM | PM | GM | GM |
| FEDAVG [69] | 80.2 | 80.2 | 85.1 | 71.6 | 71.5 | 86.9 | 50.9 | 50.2 | 76.5 | 40.1 | 40.0 | 78.2 |
| FEDPROX [59] | 81.0 | 81.0 | 82.3 | 70.4 | 70.2 | 87.0 | 58.6 | 58.7 | 76.5 | 37.6 | 38.0 | 76.6 |
| SCAFFOLD [47] | 81.1 | 81.1 | 82.2 | 72.0 | 71.8 | 86.9 | 58.7 | 58.7 | 76.6 | 38.4 | 38.4 | 77.6 |
| FEDDYN [11] | 83.3 | 83.2 | 86.4 | 77.2 | 77.1 | 87.5 | 62.5 | 62.5 | 80.4 | 46.6 | 46.5 | 80.9 |
| MOCHA [83]* | 45.3 | 45.4 | 75.6 | 47.7 | 47.5 | 77.5 | 17.9 | 18.2 | 63.4 | 14.4 | 14.9 | 65.9 |
| LG-FEDAVG [63]* | 62.8 | 62.4 | 82.4 | 74.0 | 74.1 | 83.2 | 31.2 | 31.5 | 62.8 | 24.9 | 24.8 | 66.3 |
| PER-FEDAVG [24] | 80.1 | - | 82.5 | 72.0 | - | 78.5 | 46.3 | - | 77.2 | 31.7 | - | 74.3 |
| pFEDME [20] | 78.9 | 78.9 | 81.6 | 69.3 | 69.2 | 71.6 | 46.2 | 46.2 | 54.2 | 31.7 | 31.8 | 50.6 |
| DITTO [58] | 81.0 | 81.0 | 83.7 | 71.8 | 71.6 | 86.5 | 51.0 | 50.9 | 73.1 | 40.3 | 40.2 | 75.4 |
| FedFOMO [103]* | 65.5 | 65.2 | 89.5 | 64.3 | 64.4 | 90.1 | 42.7 | 42.6 | 76.6 | 23.6 | 23.8 | 76.7 |
| Local only | - | - | 76.1 | - | - | 79.8 | - | - | 72.1 | - | - | 74.5 |

Table 8: G-FL accuracy on class-imbalanced test data. Here we use CIFAR-10 Dir(0.3).

| Method / Model | Test Set | IM10 | IM100 |
|----------------|----------|------|-------|
| FEDAVG [69] | 61.8 | 73.0 |
| FEDPROX [59] | 63.1 | 72.9 |
| SCRAFT [47] | 65.4 | 70.4 |
| FEDDYN [11] | 68.6 | 73.3 |
| FedROD | 71.9 | 76.0 |

Table 9: Main results in G-FL accuracy and P-FL accuracy (%), following Table 1 of the main paper. FedROD is compatible with other generic FL methods.

| Dataset | Non-IID | Dir(0.1) | Dir(0.3) | Dir(0.1) | Dir(0.3) |
|---------|---------|----------|----------|----------|----------|
| Test Set | G-FL | P-FL | G-FL | P-FL | G-FL | P-FL |
| Method / Model | GM | GM | PM | GM | GM | PM | GM | GM |
| FEDAVG [69] | 81.1 | 81.0 | 91.5 | 83.4 | 83.2 | 90.5 | 57.6 | 57.1 | 90.5 | 68.6 | 69.4 | 85.1 |
| FEDPROX [59] | 82.2 | 82.3 | 91.4 | 84.5 | 84.5 | 89.7 | 58.7 | 58.9 | 89.7 | 69.9 | 69.8 | 84.7 |
| SCRAFT [47] | 83.3 | 83.0 | 90.0 | 85.1 | 85.0 | 90.4 | 61.2 | 60.8 | 90.1 | 71.1 | 71.5 | 84.8 |
| FEDDYN [11] | 83.2 | 83.2 | 90.7 | 86.1 | 86.1 | 91.5 | 63.4 | 63.9 | 92.4 | 72.5 | 75.2 | 85.4 |
| FedROD (linear) | 83.9 | 83.9 | 92.7 | 86.3 | 86.3 | 94.5 | 68.5 | 68.5 | 92.7 | 76.9 | 76.8 | 86.4 |
| FedROD (hyper) | 83.9 | 83.9 | 92.9 | 86.3 | 86.3 | 94.8 | 68.5 | 68.5 | 92.5 | 76.9 | 76.8 | 86.8 |
| + FEDPROX | 83.3 | 83.3 | 92.8 | 85.8 | 85.7 | 92.3 | 70.6 | 70.5 | 92.5 | 74.5 | 74.5 | 85.7 |
| + SCRAFT | 84.3 | 84.3 | 94.8 | 88.0 | 88.0 | 94.7 | 72.0 | 71.8 | 92.6 | 77.8 | 77.8 | 86.9 |
| + FEDDYN | 85.9 | 85.7 | 95.3 | 87.3 | 87.3 | 94.6 | 68.2 | 68.2 | 92.7 | 74.6 | 74.6 | 85.6 |

personalized FL algorithm. We further replicate the experiments in [58] on robustness against adversary attacks in Table 11. Besides comparing LM and PM, we also evaluate the global model GM for P-FL accuracy. With out adversarial attacks, the LM model outperforms the PM model. However, with adversarial attacks, the PM model notably outperforms the other two models. We surmise that, when there are adversarial
Table 10: The P-FL accuracy by the two local models of Ditto [58].

| Method | FMNIST | CIFAR-10 |
|--------|--------|----------|
|        | Dir(0.1) | Dir(0.3) | Dir(0.1) | Dir(0.3) |
| PM     | 89.4    | 90.1     | 86.8    | 81.5    |
| LM     | 90.8    | 90.6     | 90.8    | 86.2    |

Table 11: Ditto with adversary attacks. We report the averaged personalized accuracy on benign clients.

| Attack                | PM | LM | GM |
|-----------------------|----|----|----|
| None                  | 94.2 | 94.7 | 91.7 |
| Label poisoning       | 93.6 | 54.5 | 84.8 |
| Random updates        | 93.2 | 54.5 | 88.7 |
| Model replacement     | 63.6 | 49.8 | 42.2 |

clients, the resulting generic model will carry the adversarial information; re-initializing the local models with it thus would lead to degraded performance.

D.8 Additional studies and discussions

Different network architectures. Fed-RoD can easily be applied to other modern neural network architectures. In Table 12, we show that Fed-RoD can be used with deeper networks.

Fed-RoD is not merely fine-tuning. Fed-RoD is not merely pre-training the model with BSM and then fine-tuning it with ERM for two reasons. First, for Fed-RoD (linear), the P-head is learned dynamically with the updating feature extractor across multiple rounds. Second, for Fed-RoD (hyper), the hypernetwork has to undergo the local training and global aggregation iterations over multiple rounds. In Table 2 of the main paper, we report the fine-tuning baseline. On CIFAR-10 Dir(0.3), it has 84.5% for P-FL (PM), lower than 86.4% and 86.8% by Fed-RoD (linear) and Fed-RoD (hyper). Note that, hypernetworks allow fast adaptation for new clients.

Comparison to the reported results in other personalized FL papers. Existing works usually report FedAvg’s personalized performance by evaluating its global model (i.e., the GM column in Table 1 of the main paper). In this paper, we evaluate FedAvg’s local model \( w_m \) (i.e., the PM column in Table 1 of the main paper), which is locally trained for epochs. We see a huge performance gap between these two models. In [24], the authors investigated a baseline “FedAvg + update”, which fine-tunes FedAvg’s global model \( \bar{w} \) with only few mini-batches for each client. The resulting personalized models thus capture less personalized information than \( w_m \) in FedAvg. For a fair comparison, we also strengthen PER-FedAvg [24] by updating with more epochs.

Table 12: Fed-RoD on CIFAR-10, Dir(0.3).

| Network       | G-FL | P-FL |
|---------------|------|------|
|               | GM   | GM   | PM  |
| ConvNet [55]  | 76.9 | 76.8 | 86.8 |
| VGG11 [82]    | 82.2 | 82.1 | 88.2 |
| ResNet8 [35]  | 80.3 | 80.0 | 86.6 |
| ResNet20 [35] | 84.0 | 83.5 | 88.5 |