

FedBug: A Bottom-Up Gradual Unfreezing Framework for Federated Learning

Chia-Hsiang Kao¹

¹National Taiwan University
chiahsiangkao.tw@gmail.com

Yu-Chiang Frank Wang^{1,2}

²NVIDIA
frankwang@nvidia.com

Abstract

Federated Learning (FL) offers a collaborative training framework, allowing multiple clients to contribute to a shared model without compromising data privacy. Due to the heterogeneous nature of local datasets, updated client models may overfit and diverge from one another, commonly known as the problem of client drift. In this paper, we propose FedBug (Federated Learning with Bottom-Up Gradual Unfreezing), a novel FL framework designed to effectively mitigate client drift. FedBug adaptively leverages the client model parameters, distributed by the server at each global round, as the reference points for cross-client alignment. Specifically, on the client side, FedBug begins by freezing the entire model, then gradually unfreezes the layers, from the input layer to the output layer. This bottom-up approach allows models to train the newly thawed layers to project data into a latent space, wherein the separating hyperplanes remain consistent across all clients. We theoretically analyze FedBug in a novel over-parameterization FL setup, revealing its superior convergence rate compared to FedAvg. Through comprehensive experiments, spanning various datasets, training conditions, and network architectures, we validate the efficacy of FedBug. Our contributions encompass a novel FL framework, theoretical analysis, and empirical validation, demonstrating the wide potential and applicability of FedBug.¹

1 Introduction

Federated Learning (FL) is a distributed approach that enables multiple clients to collaboratively train a shared model without disclosing their raw data. Federated Average (FedAvg) [27], one of the most influential FL frameworks, has served as the cornerstone for numerous algorithms in the field. Below, we provide a concise explanation of how FedAvg operates: It involves a central server and several clients. In each global round, the server disseminates the current model to all clients. Each client independently trains its model using its local data until convergence. Once the local training is completed, the clients transmit their models back to the server. The server then averages these models to produce an updated global model, which is subsequently employed in the next round.

In FL, *client drift* refers to the inconsistency between models learned by different clients, arising primarily due to the disparities in their private data distribution [8, 18, 22, 26]. As local models overfit to their datasets and converge to local minima, the global model — derived from averaging client models — compromises in terms of convergence and performance [24, 48, 49]. Extensive studies have developed strategies to tackle the client drift issue. We specifically focus on those utilizing *anchors* shared among clients, including gradient anchors and feature anchors. Gradient anchors involve the use of shared gradient information to guide the update of the client model, thereby promoting alignment and mitigating client drift [3, 17, 18, 23, 43]. On the other hand, feature anchors

¹Code is available at <https://github.com/IandRover/FedBug>

rely on shared feature information to assist in feature alignment and regularization of the feature space [26, 37, 38, 42]. However, these methods may necessitate the extra transmission of gradient information or increased regularization costs [17, 18, 23, 43] and pose privacy concerns [26, 37, 42].

To mitigate the client drift problem, another line of research has focused on leveraging model parameter anchors with enhanced training efficiency. For example, FedBABU [30] proposes to fix the classifier throughout training and update it only during evaluation. As all clients share the same fixed classifier, a set of decision boundaries is common to all clients, serving as a shared reference for updating the encoder. While FedBABU yields promising results in personalized FL scenarios, in which models are allowed to be fine-tuned using the client’s private data during the evaluation stage, FedBABU’s performance in general FL settings is less optimal and lacks theoretical understanding.

In this work, we seek to extend FedBABU by including more intermediate layers as reference points, ensuring trainability in general FL scenarios and providing theoretical support. Our approach hinges on two insights: (1) At the start of each global round of FedAvg, all clients receive an identical model from the server, and (2) each intermediate layer parameterizes hyperplanes that separate latent features. Taken together, these insights suggest a strategy: By freezing the models received from the server, we can exploit the consistency of the hyperplanes across clients to provide a common feature space for alignment.

Building on these insights, we introduce FedBug (**F**ederated Learning with **B**ottom-**U**p **G**radual Unfreezing), a novel FL framework leveraging shared parameter anchors to mitigate client drift. Unlike FedAvg, FedBug begins local training by freezing the entire model, then gradually thaws the layers from the input layer to the output layer. The key mechanism operates as follows: when a layer becomes trainable (thawed) while its succeeding layers remain frozen, this thawed layer learns to project its inputs into a shared feature space. This space is notably defined by the hyperplanes of the still-frozen succeeding layers, providing a common reference across clients and enhancing cross-client alignment. Then, FedBug progressively unfreezes the next layer, ensuring the layer’s trainability after it has served as a shared reference. This results in a framework that balances alignment and adaptability.

We conduct a thorough investigation of FedBug through theoretical analysis and extensive experiments. Notably, we are the first to analyze FL algorithms in an over-parameterized setting, pushing the boundaries of the current FL theoretical framework. Utilizing a two-layer linear network with an orthogonal regression task setting, our analysis reveals that FedBug converges faster than FedAvg. Additionally, we conduct extensive experiments across various datasets (CIFAR-10, CIFAR-100, Tiny-ImageNet), training conditions (label skewness and client participation rate), and network architectures (simple CNN, Resnet-18, Resnet34) to ensure the broad applicability of FedBug.

Our contributions are three-fold:

- **Novel Federated Learning Framework:** We propose FedBug, a novel federated learning framework that leverages model parameters as anchors to effectively address the challenges of client drift in federated learning.
- **Theoretical Analysis:** We provide a theoretical analysis of the FedBug framework, demonstrating its better convergence rate compared to FedAvg. Our analysis focuses on a two-layer linear network with an orthogonal regression task, offering novel insights into federated learning dynamics in the context of over-parameterized models.
- **Empirical Validation:** We extensively validate the effectiveness of FedBug through a series of experiments on diverse datasets (CIFAR-10, CIFAR-100, Tiny-ImageNet), varying training conditions (label skewness, client participation rate), and different model architectures (standard CNN, ResNet18, ResNet34). Furthermore, we assess the compatibility of the FedBug framework with other federated learning algorithms. Our empirical findings underscore the immense potential and wide-ranging applicability of FedBug.

2 Literature Review

Mitigating Client Drift Using Gradient and Feature Anchors. To address the client drift problem in federated learning, various methods have been explored to explicitly or implicitly provide shared reference points for client models. **Gradient anchors** are employed in several FL methods. For

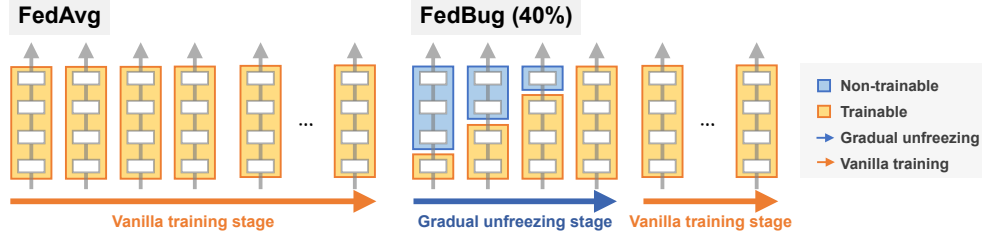


Figure 1: **Comparisons of FedAvg and FedBug.** While FedAvg updates all network modules during local training, FedBug gradually unfreezes network modules in a bottom-up manner to tackle client drift. Take FedBug (40%) for example, the first 40% of training iterations perform gradual unfreezing (GU), while the remaining 60% perform vanilla training. With the same number of training iterations, FedBug has fewer parameters to update and thus exhibits improved learning efficiency.

instance, SCAFFOLD [18] incorporates server-level gradients in updates to reduce local noise effects, while FedCM [43], FedGLOMO [3], MIME [17], and FedDANE [23] leverage server-level gradient to align clients by providing mutual update directions. While these approaches have demonstrated effectiveness, they require sharing additional gradient information with the local clients. **Feature anchors** are used in methods such as CCVR [26], VHL [38], FedProto [37], and FedFA [42]. These methods employ outside datasets or clients’ private datasets to produce and regularize features across different clients. However, these approaches may suffer from privacy leakage, increased dataset or feature transmission costs, and added computation budget.

Mitigating Client Drift Using Parameter Anchor. Recent research has emphasized the prominence of client drift in the top layers of models. Specifically, [8, 22, 26, 49] demonstrate that the penultimate layer and classifier exhibit the lowest feature similarities among the clients. These findings suggest that local classifiers undergo significant changes to adapt to the local data distribution, which amplifies challenges related to class imbalance and results in biased model predictions for specific classes [8, 20, 35, 48]. This phenomenon is consistent with research on the long-tail analysis, where [16, 46] demonstrated that the head is biased in class-imbalanced environments. To explicitly address the issue of client drift, certain studies leverage parameter anchors to align the clients. FedProx [24] regularizes the L2 distance between the client model and server model, establishing a shared reference in the parameter space. Additionally, FedBABU [30] falls into this category as it uses a fixed classifier as the parameter-level anchor during training. As each client’s frozen classifier parameterizes the same set of decision boundaries in the feature space, FedBABU also serves as a hyperplane anchor in the latent space. However, FedBABU’s performance is suboptimal and necessitates additional personalized training (fine-tuning) on the clients’ private dataset during the evaluation stage, a process known as personalization, to achieve satisfactory results.

3 Method

FedBug is designed to pivot parameter anchors to reduce client drift. As shown in Figure 1, FedBug divides the local training phase into the gradual unfreezing (GU) stage and the vanilla training stage. In the vanilla training stage, all modules of a model are trainable. During the GU stage, FedBug freezes the whole model initially and then progressively unfreezes its modules bottom-up. This stage is equally divided into several periods, with an additional module being unfrozen in each period.

We first introduce the notations used in this work. Consider N clients, where each client i has its own loss function F_i . The model is denoted as θ and is composed of m modules, where a *module* can refer to a single CNN layer or a ResNet block. For simplicity, we use $\theta^{1:m}$ to signify the first m modules of the model, with θ^1 indicating the input module. We define P as the percentage of the GU stage and K as the total number of local training iterations, hence the GU stage spans PK local iterations. We denote η_g and η_l as global and local learning rates, respectively. Algorithm 1 provides a detailed description of the FedBug framework. In line seven of the algorithm, the variable m is determined based on the current local iteration step k and specifies the number of modules to be updated. For a comprehensive comparison between FedAvg and FedBug, please refer to Appendix C.

We now explore the underlying rationale of FedBug, using Figure 1 for illustration. Consider a four-module model trained using FedBug (40%), represented as $M = 4$, and $P = 0.4$. In this case, a module becomes trainable every $0.1K$ local iteration. Suppose we are in the second GU period, where all clients have just unfrozen their second module. During this period, the clients adapt their first and second modules and project the data into a feature space. Notably, the separating hyperplanes within this feature space are parameterized by the yet-to-be-unfrozen modules (the third and fourth modules in this case). These modules remain consistent during this period, serving as a shared anchor among clients. Similarly, as we progress to the subsequent third period, this process continues, with clients mapping their data into decision regions defined by the still-frozen fourth module. By leveraging the shared reference, FedBug ensures ongoing alignment among the clients.

Algorithm 1 FedBug: Federated Learning with Bottom-Up Gradual Unfreezing

Notation:

- $\theta^{1:m}$: the first m modules of model θ
 - R : number of global rounds
 - K : number of local iterations
 - P : gradual unfreezing stage percentage
 - 1: **Input:** global model θ with M modules
 - 2: **for** $r = 1, \dots, R$ **do**
 - 3: Sample clients $S \subseteq \{1, \dots, N\}$
 - 4: **for** each client $i \in S$ in parallel **do**
 - 5: Initialize local model $\theta_i \leftarrow \theta$
 - 6: **for** $k = 1, \dots, K$ **do**
 - 7: $m \leftarrow \min\{M, \lceil \frac{kM}{PK} \rceil\}$
 - 8: // Update m modules of θ_i
 - 9: $\theta_i^{1:m} \leftarrow \theta_i^{1:m} - \eta_l \nabla F_i(\theta_i^{1:m})$
 - 10: **end for**
 - 11: $\Delta_i \leftarrow \theta_i - \theta$
 - 12: **end for**
 - 13: $\theta \leftarrow \theta + \frac{\eta_g}{|S|} \sum_{i \in S} \Delta_i$
 - 14: **end for**
-

4 Theoretical Analysis

In this section, we present a theoretical analysis of the convergence of FedBug. We start with a brief overview, explaining the motivation behind our theoretical setup. We then discuss the technical challenges and elaborate on the key intuitions guiding our approach. Finally, we present the results of our analysis, which offer valuable insights into the convergence behavior of FedBug within the theoretical framework we have considered.

Motivation 1: Extending From Single-Variable Loss Function. As our FedBug leverages a shared feature space across clients during local iterations, an theoretical understanding of it requires at least two neural layers so that the concept of feature space is valid. However, conventional FL theoretical frameworks often restrict to single-variable loss functions for clients, which may not adequately capture the dynamics of FedBug. To bridge this gap, we introduce an over-parameterized analytic FL framework that provides a more comprehensive understanding of the FL dynamics.

Motivation 2: Addressing Client Drift via Dataset Heterogeneity. While previous FL analyses mainly tackle label distribution [23, 28], clients' loss functions [1, 18, 43], and local training conditions [41] as sources of client drift, limited attention has been given to client drift caused by dataset distributions. Thus, we aim to explore the impact of dataset domain heterogeneity on client drift. This heterogeneity becomes apparent in over-parameterized setups, where the models have sufficient capacity to capture different aspects of the data distributions across clients.

Theoretical Setup and Technical Challenges. Our analysis centers on regression tasks with linear neural network [2, 6, 7, 15, 34, 39, 44], focusing specifically in an orthogonal dataset distribution. This particular setup has been studied in the context of transfer learning with out-of-distribution dataset [19] and domain generalization [21]. For more detailed information about the task setup, please refer to Appendix B.1. Analyzing FL algorithms in the context of a two-layer linear network with an orthogonal dataset distribution presents several challenges. As we show in Appendices B.3 and B.4, over-parameterization results in non-convexity [33] and hyperbolic trajectories of local parameters update, making the analysis difficult. Moreover, as we focus on the FL setup, the convergence of server model parameters must be considered, further complicating the problem.

Intuition and Assumptions. To tackle these challenges, we follow [21] and consider a simplified scenario with two clients. Additionally, we propose three assumptions: (1) linear approximation of the global minimum, (2) bounding of local convexity, and (3) linear approximation of local learning trajectories. These assumptions are akin to those utilized in previous FL theoretical frameworks, such as bounded gradient norm, convexity, smoothness, cross-client gradient, or Hessian dissimilarity [17, 23, 24, 32, 40, 43]. They serve to streamline the analysis process while capturing essential aspects of

convergence speed characterization. Although the proposed assumptions may not fully capture the complexity of the problem, they provide a valuable starting point for understanding the dynamics of FL in over-parameterized models.

Theoretical Results. Our analysis demonstrates that, while FedAvg converges linearly to a global minimizer, its convergence is limited by a mismatch between the global and client update directions. In contrast, our FedBug introduces a single parameter freezing step that encourages greater alignment between the local and global update directions, leading to significantly improved convergence. It offers a theoretical foundation for the effectiveness of the FedBug algorithm and sheds light on the dynamics of FL in this challenging setup.

In the following subsections, we detail the task and model setup for our analysis in Section 4.1, present the key theoretical analysis in Section 4.2, and describe the simulation experiments in Section 4.3. For a comprehensive theoretical analysis, please refer to Appendix B.

4.1 Task Setting and Model Architecture

To ensure tractability in characterizing FedBug, we focus on a two-layer linear network as the FL model f , with an orthogonal dataset distribution as a source of client drifts. We consider a FL regression task, with two clients denoted as c_1 and c_2 . Each client has different regression data, specifically $\mathcal{T}_1 = \{x_1 = [1, 0], y_1 = 1\}$ and $\mathcal{T}_2 = \{x_2 = [0, 1], y_2 = 1\}$. The objective is to minimize the L2 loss, with client c_1 (c_2) minimizing $L_1 = \|f(x_1) - y_1\|$ ($L_2 = \|f(x_2) - y_2\|^2$). Note that the network f is with two nodes $[a, b]$ in the first layer and one node $[v]$ in the second layer. Specifically, $f(x) = x[a, b]^\top v$. The task setup implies that client c_1 (c_2) aims to minimize $L_1 = |av - 1|^2$ ($L_2 = |bv - 1|^2$).

4.2 Convergence Rates of FedAvg and FedBug

In order to analyze the convergence properties of both algorithms, we focus on the region around the minima and make additional assumptions for tractability. These assumptions characterize the landscape near the server model, local client models, and global minima.

Assumption 1. (Local Linearity) *The server model parameters are initialized in the vicinity of the global minimum, such that the global minimum can be locally approximated as a linear function.*

Assumption 2. (Bounded Local Convexity) *Under Assumption 1, there exist constants $\beta_1 > 0$ and $0 < \beta_2 < 1$, such that for all n , the value of v^{n^2} is bounded as follows: $\beta_1\beta_2 \leq v^{n^2} \leq \beta_1$.*

Assumption 3. (Orthogonal Trajectory and Bounded Error) *Under Assumptions 1 and 2, with a properly chosen step size, the local gradient descent update trajectory can be approximated by a linear trajectory orthogonal to the global minimum. The approximation error, which quantifies the discrepancy between the gradient descent solution and the ideal orthogonal linear path solution, is bounded by α times the length of the orthogonal linear trajectory.*

Assumption 1 ensures that FedAvg and FedBug starts close enough to the global minimum, enabling convergence without the interference of poor initialization and unbounded convexity. Building on Assumption 1, Assumption 2 further bound the local convexity around the global minimum, so that both FedAvg and FedBug are prevented from diverging or oscillating around the global minimum when using an appropriate step size. Lastly, Assumption 3 guarantees that FedBug converges along a favorable direction and helps us to quantify the improvement brought by a single global round while allowing for quantification of the approximation error. Please refer to the right panel of Figure 5 for illustration. We defer the discussion of limitation to Section B.7.

Distinct from conventional FL convergence analysis, we discover that our unique setup permits an alternative approach for proving the convergence behavior of FedAvg. The cornerstone of our theorem is the observation that at each global round, the discrepancy between clients' parameters diminishes by a factor of r . Here, r is determined by the degree of alignment between the local update and the axial direction of the last layer parameters — a measure that captures the consistency between the local and the global updates.

To measure this discrepancy reduction ratio, we define two useful terms:

Definition 1. *Client discrepancy d^i is the L1 distance between the server model parameters a^i and b^i at the i -th global round: $d^i = \|a^i - b^i\|$.*

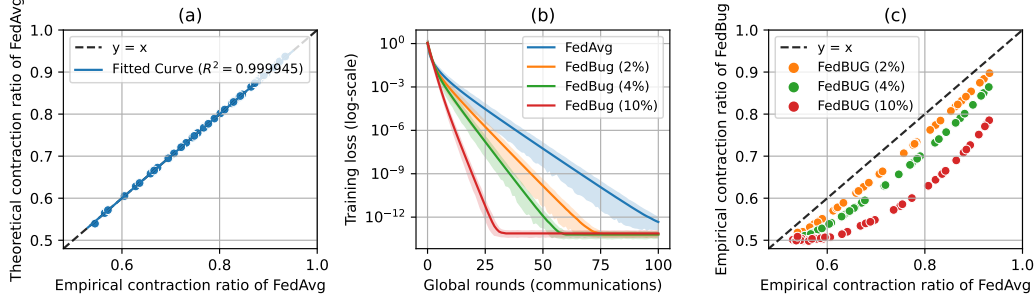


Figure 2: (a) Alignment between theoretical and empirical client discrepancy contraction ratios. (b) Comparisons of training loss curves of FedAvg and FedBug. (c) Comparison of the contraction ratio of FedAvg and FedBug. Note that (a) is shown to confirm the validity of our assumptions and theorems, (b) is to validate our improved convergence over FedAvg, and (c) is to further verify that FedBug achieves reduced client discrepancy when comparing to FedAvg.

Definition 2. Client discrepancy contraction ratio r : $r = d^{i+1}/d^i$.

Now, we present the theorem describing the convergence behavior of FedAvg:

Theorem 1. Under Assumptions 1, 2 and 3, models trained using FedAvg converge with the client discrepancy contraction ratio upper bounded by $\frac{1+\cos^2 \theta(1+\alpha)}{2}$, where θ is the angle between the local update direction and the axis of the last layer parameter.

Theorem 2. Under Assumptions 1, 2, and 3, there exists $1 - \beta_2 < m < 1$ such that if the step size satisfies $\frac{1-m}{\beta_1\beta_2} < \eta < \frac{1}{\beta_1}$, FedBug converges with a client discrepancy contraction ratio upper bounded by $\frac{1+m\cos^2 \theta(1+\alpha)}{2}$, where θ is defined as in Theorem 1.

Proof Sketch. We defer the complete proof to Appendix B.5 and provide a proof sketch here. We start with the case of FedAvg. The goal of the proof is to show the discrepancy between clients contracts after each global round. To estimate the discrepancy term, we make use of geometric considerations and find that the client’s discrepancy after local updates is a double projection of the original client discrepancy, where the projection angle is the difference between the local update direction and the axis of the last layer parameter. As for the convergence of FedBug, we analyze the case where the first layer parameters are updated for one step, while the final layer parameter is kept frozen. With a proper step size, this update directly minimizes the client discrepancy and thus leads to a lower client discrepancy when the clients reach their local minimum, compared with FedAvg. As a result, we can show that FedBug exhibits a faster convergence speed than FedAvg.

4.3 Empirical Validation of Convergence Rate

We conduct simulation experiments based on our task and model setup to empirically verify our theorem and the underlying assumptions. Details over setup can be found in Appendix B.6. The results are presented in Figure 2. In Fig. 2(a), we observe a high correlation between the contraction ratio estimated theoretically using θ and the one obtained by following Definitions 1 and 2 in FedAvg. This supports the validity of our assumptions and theorems. Based on the training loss curves shown in Fig. 2(b), we see that FedBug converges faster than FedAvg, confirming its superior convergence properties. As for Fig. 2(c), we compare the empirical client discrepancy contraction ratios between FedAvg (x-axis) and FedBug (y-axis). Given a contraction ratio of FedAvg, further decreased ratios can be observed for FedBug with varying GU ratios. These simulation experiments provide empirical evidence supporting our theoretical analysis and verify the improved performance of FedBug compared to FedAvg.

5 Experiments

We present an extensive evaluation of FedBug across various FL scenarios. We provide a brief overview of the datasets and models, with a detailed description of our setup available in the

CIFAR-10 (# clients: 100; client participation rate: 1%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.3$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	78.61	79.65	80.62	80.03	80.15	76.13	77.06	78.17	77.14	76.80
FedBug (10%)	80.22	79.92	81.02	80.74	79.96	77.98	77.89	78.78	77.94	77.58
FedBug (20%)	80.63	80.31	81.47	81.00	80.19	77.68	77.67	78.80	77.73	77.22
FedBug (40%)	80.43	80.34	81.06	80.70	80.27	77.54	77.54	78.36	77.45	77.31

CIFAR-10 (# clients: 100; client participation rate: 10%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.3$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	80.06	79.86	82.62	79.95	80.01	78.11	77.67	81.15	77.74	78.12
FedBug (10%)	80.57	80.51	82.89	80.69	80.66	78.43	78.58	81.21	78.27	78.65
FedBug (20%)	80.66	80.55	83.27	81.10	81.31	78.86	78.77	81.48	78.86	78.67
FedBug (40%)	80.91	81.02	83.60	81.26	81.37	79.18	79.28	81.80	79.05	79.20

Table 1: **Experiments on CIFAR-10 with standard CNN.** We conduct experiments with different client participation rates (1% and 10%), degrees of heterogeneity ($\alpha \in \{0.3, \infty\}$), and combinations of FL algorithms. Test accuracy (%) is averaged over four seeds in all experiments. Bold font indicates the highest accuracy in each column, while the gray background highlights our framework.

CIFAR-100 (# clients: 100; client participation rate: 1%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.3$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	39.33	40.19	49.61	39.51	42.71	40.43	40.46	48.69	41.32	43.81
FedBug (10%)	41.56	41.17	51.18	42.26	43.38	42.99	42.27	50.02	42.86	44.53
FedBug (20%)	43.28	43.49	51.49	43.51	44.14	43.47	43.34	50.58	43.67	44.75
FedBug (40%)	45.59	45.33	51.91	45.83	44.87	45.35	44.61	51.00	45.03	44.78

CIFAR-100 (# clients: 100; client participation rate: 10%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.3$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	44.45	44.64	46.62	46.01	45.14	44.05	43.58	45.01	44.89	43.28
FedBug (10%)	47.51	46.85	47.89	48.68	46.12	47.38	47.07	47.03	47.83	44.27
FedBug (20%)	48.77	48.35	48.20	49.54	46.30	47.92	47.87	47.63	48.16	43.99
FedBug (40%)	49.51	49.42	48.80	50.03	46.29	47.80	47.67	47.32	47.87	44.02

Table 2: **Experiments on CIFAR-100 with standard CNN.** We conduct experiments with different client participation rates (1% and 10%), degrees of heterogeneity ($\alpha \in \{0.3, \infty\}$), and FL algorithms. Note that improvements on CIFAR-100 are more remarkable than those on CIFAR-10.

Appendix. The code is written in PyTorch and executed on a single GPU, either an NVIDIA 3090 or V100. All experiments are performed with four distinct random seeds. Experimental details along with the ablation study are deferred to Appendices A.1 and A.2, respectively.

5.1 Experimental Setup

Datasets. We utilized benchmark datasets that follow the same train/test splits as previous works [1, 24, 27]. These datasets include CIFAR-10, CIFAR-100, and Tiny-ImageNet. We randomly assigned data to the clients for the IID label distribution split [1, 27]. As for the non-IID label distribution, we followed the Dirichlet distribution $\text{Dir}(\alpha)$, as in [1, 47]. Here, α is a concentration parameter, with a smaller α indicating stronger data heterogeneity. When α equals ∞ , the setting is homogeneous. We set α to 0.3 for CIFAR-10 and CIFAR-100, and 0.5 for Tiny-ImageNet.

Models. For standard CNN, we employ a standard convolutional neural network, similar to [1, 27], consisting of two (three) convolutional layers followed by three fully connected layers for CIFAR-10 and CIFAR-100 (Tiny-ImageNet) dataset. For ResNet-18 and ResNet-34 [9], we change the batch normalization to group normalization [1, 10, 12, 13, 45].

Tiny-ImageNet (# clients: 10; client participation rate: 10%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.5$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	28.40	28.41	29.84	29.39	28.15	26.40	26.76	28.46	27.44	26.51
FedBug (12%)	28.87	28.92	30.18	29.76	28.77	27.23	27.34	28.94	27.93	26.76
FedBug (24%)	29.04	29.35	30.39	30.63	28.70	27.78	27.25	29.12	28.37	26.93
FedBug (48%)	29.58	29.67	27.78	30.42	29.29	27.65	27.83	29.45	28.44	27.75

Tiny-ImageNet (# clients: 10; client participation rate: 30%)										
Method	IID label distribution ($\alpha = \infty$)					Non-IID label distribution ($\alpha = 0.5$)				
	FedAvg	FedProx	FedDyn	FedExp	FedDecorr	FedAvg	FedProx	FedDyn	FedExp	FedDecorr
Vanilla	26.48	26.62	31.35	26.79	28.96	25.09	25.40	29.78	24.98	26.57
FedBug (12%)	26.93	26.95	31.31	26.79	29.50	25.69	25.71	29.95	25.72	26.52
FedBug (24%)	26.94	27.15	31.54	27.10	29.43	25.82	26.02	30.05	25.62	27.49
FedBug (48%)	27.64	27.84	31.44	27.76	29.60	26.36	26.81	30.71	26.33	27.82

Table 3: **Experiments on Tiny-ImageNet of 10 clients with standard CNN.** We conduct experiments with different client participation rates (10% and 30%), degrees of heterogeneity ($\alpha \in \{0.3, \infty\}$), and combinations of FL algorithms.

5.2 Experimental Results

Improved Performance and High Compatibility With Various FL Algorithms. The main results on CIFAR-10, CIFAR-100, and Tiny-ImageNet dataset are presented in Tables 1, 2, and 3. These results highlight the high compatibility of the FedBug framework when combined with different FL algorithms. Additionally, we consistently observe that the FedBug framework outperforms the vanilla training framework, even when the gradual unfreezing stage comprises only ten percent of the training process. This indicates the effectiveness and efficiency of the FedBug approach in improving model performance. Furthermore, the FedBug training framework demonstrates a consistent synergistic effect across five different FL algorithms, two client participation levels, and both IID and non-IID label distributions. This demonstrates the broad applicability of our proposed framework in combination with existing FL training algorithms and experimental setups.

Applicability of FedBug on ResNet. When applying FedBug to larger models like ResNet, a natural question arises: What should be the smallest unit to unfreeze during the GU stage — a ResNet Module or a residual block? Since both ResNet-18 and ResNet-34 can be seen as having four ResNet Modules or consisting of eight and sixteen residual blocks, respectively, a robust framework should remain agnostic to the unit definition. Therefore, we evaluate the adaptability of FedBug using two distinct unfreezing strategies: (1) progressively unfreezing one ResNet Module at a time, and (2) progressively unfreezing one residual block at a time. Notably, we capitalize the term ResNet "Module" to differentiate it from the general module mentioned in Algorithm 1.

The experimental results, presented in Table 4, consistently demonstrate the superiority of the FedBug framework over the vanilla training framework across both unfreezing strategies and different label distributions. Interestingly, we observe that both strategies perform comparably well on the datasets, indicating the effectiveness of FedBug.

Impact of Gradual Unfreezing Percentage. We investigate the impact of the percentage of the GU stage in CIFAR-100 and Tiny-ImageNet in the standard CNN model. The baseline framework is FedAvg, represented by a percentage of 0%. The results are shown in Figure 3. Our experiments reveal consistent improvements in test accuracy even with longer GU ratios. Notably, allocating a larger percentage of the training period to GU leads to the top layers receiving less training. Consequently, the test accuracy does not necessarily increase monotonically with the GU ratio. For instance, when training a five-layer model with FedBug using a 100% GU stage ratio, the penultimate linear layer and the classifier are trained for only 40% and 20% of the total training iterations, respectively. In this extreme scenario, FedBug not only saves considerable training time but also provides improvement. These results highlight the robustness of FedBug to the GU ratio and suggest a small portion of the training time for gradual unfreezing may readily yield favorable results.

CIFAR-100 (# clients: 10; client participation rate: 10%)								
Method	IID label Distribution ($\alpha = \infty$)				Non-IID label Distribution ($\alpha = 0.3$)			
	ResNet-18		ResNet-34		ResNet-18		ResNet-34	
	Module(4)	Block(8)	Module(4)	Block(16)	Module(4)	Block(8)	Module(4)	Block(16)
Vanilla	52.59	52.59	52.64	52.64	49.04	49.04	48.69	48.69
FedBug (20%)	53.25	53.05	53.01	53.42	49.70	49.64	49.20	49.17
FedBug (40%)	53.65	53.49	53.56	53.56	49.36	49.69	49.37	49.33

Tiny-ImageNet (# clients: 10; client participation rate: 10%)								
Method	IID label Distribution ($\alpha = \infty$)				Non-IID label Distribution ($\alpha = 0.5$)			
	ResNet-18		ResNet-34		ResNet-18		ResNet-34	
	Module(4)	Block(8)	Module(4)	Block(16)	Module(4)	Block(8)	Module(4)	Block(16)
Vanilla	33.88	33.88	33.22	33.22	31.91	31.91	31.53	31.53
FedBug (20%)	34.25	34.31	34.28	34.36	32.29	32.32	32.33	32.31
FedBug (40%)	35.28	35.17	35.12	35.10	32.86	33.47	33.20	33.36

Table 4: **Experiments on ResNet with module-wise and block-wise unfreezing strategies.** Note that two unfreezing strategies are considered: (1) unfreezing one ResNet *Module* at a time and (2) unfreezing one residual *Block* at a time. Module (4) indicates that the model consists of four ResNet Modules, while Block (16) signifies the model consists of sixteen residual Blocks. Consistent improvements of FedBug with both unfreezing strategies can be observed.

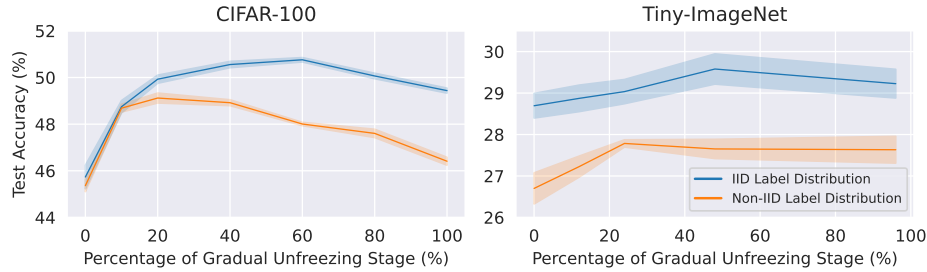


Figure 3: **Impact of Gradual Unfreezing Percentage.** We investigate the effect of the percentage of the GU stage in CIFAR-100 and Tiny-ImageNet. The baseline framework is FedAvg (i.e., GU percentage of 0.0). Our experiments reveal consistent improvements in test accuracy even with longer GU ratios, where the top layers receive considerably less training compared to FedAvg.

6 Conclusion

In this work, we introduce FedBug, a novel FL framework designed to mitigate client drift. By leveraging model parameters as anchors, FedBug aligns clients while improving learning efficiency. We perform theoretical analysis in an over-parameterized setting, revealing that FedBug achieves a faster convergence rate compared to the widely adopted FedAvg framework. To empirically validate the effectiveness, we conduct extensive experiments on various datasets, training conditions, and network architectures, consistently demonstrating the superiority and compatibility of FedBug. Overall, our contributions include the introduction of a novel FL framework, theoretical analysis, and comprehensive empirical validation, highlighting the broad potential and applicability of FedBug.

Limitations. Our analysis is limited to a specific scenario involving two clients, a two-layer linear network, and an orthogonal dataset distribution. We do not address the challenges related to multiple clients, non-IID label distributions, non-orthogonal dataset distributions, or general convex functions, leaving these as potential areas for future research. In Appendix B.7, we provide a detailed discussion on the three assumptions made in order to make the problem more tractable.

Broader Impact. The FedBug framework has the potential for wide-ranging impact in areas where FL intersects with natural language processing, reinforcement learning, autonomous driving, personalized computing, and more. The uniqueness of the bottom-up gradual unfreezing concept may also open avenues in general distributed computing.

References

- [1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations (ICLR)*, 2021.
- [2] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning (ICML)*, 2018.
- [3] Rudrajit Das, Anish Acharya, Abolfazl Hashemi, Sujay Sanghavi, Inderjit S Dhillon, and Ufuk Topcu. Faster non-convex federated learning via global and local momentum. In *Uncertainty in Artificial Intelligence*. PMLR, 2022.
- [4] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [5] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.
- [6] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [7] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [8] Yongxin Guo, Tao Lin, and Xiaoying Tang. Fedaug: Reducing the local learning bias improves federated learning on heterogeneous data. *arXiv preprint arXiv:2205.13462*, 2022.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [10] Hossein Hosseini, Hyunsin Park, Sungrack Yun, Christos Louizos, Joseph Soriaga, and Max Welling. Federated learning of user verification models without sharing embeddings. In *International Conference on Machine Learning (ICML)*, 2021.
- [11] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.
- [12] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2020.
- [13] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.
- [14] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedexp: Speeding up federated averaging via extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.
- [15] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.
- [16] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020.
- [17] 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.

- [18] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning (ICML)*, 2020.
- [19] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*, 2022.
- [20] Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. Preservation of the global knowledge by not-true distillation in federated learning. *arXiv preprint arXiv:2106.03097*, 2021.
- [21] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *International Conference on Learning Representations (ICLR)*, 2023.
- [22] Bo Li, Mikkel N Schmidt, Tommy S Alstrøm, and Sebastian U Stich. Partial variance reduction improves non-convex federated learning on heterogeneous data. *arXiv preprint arXiv:2212.02191*, 2022.
- [23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231. IEEE, 2019.
- [24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
- [25] Chen Cecilia Liu, Jonas Pfeiffer, Ivan Vulić, and Iryna Gurevych. Improving generalization of adapter-based cross-lingual transfer with scheduled unfreezing. *arXiv preprint arXiv:2301.05487*, 2023.
- [26] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [27] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.
- [28] Mahdi Morafah, Saeed Vahidian, Chen Chen, Mubarak Shah, and Bill Lin. Rethinking data heterogeneity in federated learning: Introducing a new notion and standard benchmarks. *arXiv preprint arXiv:2209.15595*, 2022.
- [29] Subhabrata Mukherjee and Ahmed Hassan Awadallah. Distilling bert into simple neural networks with unlabeled transfer data. *arXiv preprint arXiv:1910.01769*, 2019.
- [30] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.
- [31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- [32] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.
- [33] Itay M Safran, Gilad Yehudai, and Ohad Shamir. The effects of mild over-parameterization on the optimization landscape of shallow relu neural networks. In *Conference on Learning Theory*, pages 3889–3934. PMLR, 2021.
- [34] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

- [35] Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. *arXiv preprint arXiv:2204.13399*, 2022.
- [36] Yujun Shi, Jian Liang, Wenqing Zhang, Vincent YF Tan, and Song Bai. Towards understanding and mitigating dimensional collapse in heterogeneous federated learning. *arXiv preprint arXiv:2210.00226*, 2022.
- [37] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.
- [38] Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xin He, Bo Han, and Xiaowen Chu. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *International Conference on Machine Learning*, pages 21111–21132. PMLR, 2022.
- [39] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning (ICML)*, 2021.
- [40] Qianqian Tong, Guannan Liang, and Jinbo Bi. Effective federated adaptive gradient methods with non-iid decentralized data. *arXiv preprint arXiv:2009.06557*, 2020.
- [41] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.
- [42] Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. In *International Conference on Learning Representations (ICLR)*, 2023.
- [43] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. Fedcm: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.
- [44] Haotian Ye, James Zou, and Linjun Zhang. Freeze then train: Towards provable representation learning under spurious correlations and feature noise. In *International Conference on Artificial Intelligence and Statistics*, 2023.
- [45] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2066–2074, 2021.
- [46] Haiyang Yu, Ningyu Zhang, Shumin Deng, Zonggang Yuan, Yantao Jia, and Huajun Chen. The devil is the classifier: Investigating long tail relation classification with decoupling analysis. *arXiv preprint arXiv:2009.07022*, 2020.
- [47] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning (ICML)*, pages 7252–7261, 2019.
- [48] Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning (ICML)*, 2022.
- [49] 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.

A Experiments

A.1 Implementation Details

Standard CNN. Our code is based on [1] and we extend it to include the Tiny-ImageNet dataset, the latest proposed FedDecorr [36] and FedExp [14] algorithms. We use Stochastic Gradient Descent (SGD) optimizer and a cross entropy loss function, with a learning rate of 0.1 and weight decay of 0.001. We use a batch size of 50 and perform horizontal flipping for training data augmentation on all datasets, while adding cropping augmentation on CIFAR-10 and CIFAR-100 [1]. For the training epochs, we run 300, 500, and 100 global rounds (communication rounds) with 10, 5, and 5 local epochs in CIFAR-10, CIFAR-100, and Tiny-ImageNet, respectively, as suggested in [1, 36]. In the CIFAR-10 and CIFAR-100 (Tiny-ImageNet) datasets, we consider 100 (10) participants with participation rates of 1% and 10% (10% and 30%) at each global round, respectively [1]. 1% participation rate means that each client had a 1% chance of being selected to join in a global round.

ResNet. For the ResNet architecture, we focus on CIFAR-100 and Tiny-ImageNet datasets. We conducted 100 global rounds with 5 local epochs on CIFAR-100 and 3 local epochs on Tiny-ImageNet [1, 36]. In both datasets, we work with 10 participants with a 10% client participation rate. Notably, ResNet-18 and ResNet-34 can be treated as having either four modules each, or eight and sixteen residual blocks, respectively. To test the applicability of FedBug, we consider two strategies: (1) unfreezing one ResNet module at a time, or (2) unfreezing one residual block at a time. The first strategy corresponds to a scenario where $M = 4$, while the second corresponds to $M = 8$ for ResNet-18 and $M = 16$ for ResNet-34.

Hyperparameters. Our use of hyperparameters is similar to [1], where $\mu = 0.0001$ for FedProx [24], $\alpha = 0.01$ for FedDyn [1]. We use $\beta = 0.01$ for FedDecorr [36], while FedExp [14] does not require additional hyperparameters.

A.2 Experimental Results

Ablation Study: Comparative Analysis of Freezing Strategies. To compare the impact of different freezing strategies, we include the following methods that are closely related to ours: (1) Top-Down Gradual Unfreezing, which has been adopted in recent NLP literature for model fine-tuning [11, 25, 29, 31] and fine-tuning the model from the output layer to the input layer; (2) Fixing the Last Layer throughout training, known as FedBABU [30]; and (3) Fixing the Last Two Layers, i.e., the classifier and the penultimate layer. It is worth noting that in the work of [[11]], they propose the use of Universal Language Model Fine-tuning (ULMFiT) for fine-tuning large NLP models for transfer learning, where the method incorporates top-down gradual unfreezing. The rationale behind top-down gradual unfreezing is that the last layer contains the least general knowledge and is more specialized for the original task. By employing top-down gradual unfreezing, the bottom layers undergo fewer changes, mitigating the risk of catastrophic forgetting and facilitating adaptation to the new task. While ULMFiT focuses on specific task adaptation for fine-tuning, the aim of FedBug is to increase cross-client alignment through parameter anchors.

For our baseline, we utilize FedAvg and FedBug (20%), referred to as "Vanilla" and "FedBug: Bottom-Up GU", respectively. The results on CIFAR-10, CIFAR-100, and Tiny-ImageNet are shown in Figure 4. We observe that all the alternative unfreezing or freezing frameworks perform worse than FedAvg, while our proposed method consistently outperforms them. This ablation study highlights the importance of freezing strategies in achieving optimal test performance and provides compelling evidence supporting the intuition that gradually unfreezing layers from the bottom up leads to improved performance.

Ablation Study: Number of Clients. To compare the impact of different numbers of clients, we conducted an ablation study using a consistent client participation rate of 10% for each setting. We utilized the CIFAR-100 dataset with ResNet-18 and employed the ResNet Module-wise unfreezing strategy. We utilize FedBug (50%) for this ablation study. The experimental results were averaged over four random seeds. The results are summarized in Table 5, revealing that even with a large number of clients, the FedBug framework consistently improves testing accuracy.

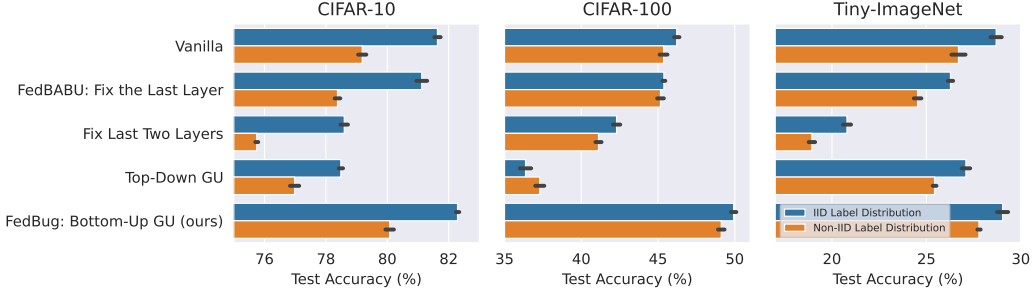


Figure 4: **Impact of Gradual Unfreezing Percentage.**

CIFAR-100 (ResNet-18; client participation rate: 10%)						
Method	IID label distribution ($\alpha = \infty$)			Non-IID label distribution ($\alpha = 0.3$)		
	# Clients			# Clients		
	10	50	500	10	50	500
Vanilla	52.55	42.40	19.45	49.05	40.65	19.32
FedBug	53.59	44.44	21.79	49.93	41.32	19.56

Table 5: **Experiments on CIFAR-100 with varying number of clients on ResNet-18.**

B Full Theoretical Analysis

In this section, we present a theoretical analysis of the FedAvg and FedBug algorithms in a two-layer linear network with an orthogonal dataset distribution.

B.1 Task Setting and Model Architecture

Task and Evaluation. We consider a FL regression task, with two clients denoted as c_1 and c_2 . Each client has different regression data, specifically $\mathcal{T}_1 = \{x_1 = [1, 0], y_1 = 1\}$ and $\mathcal{T}_2 = \{x_2 = [0, 1], y_2 = 1\}$. The objective is to minimize the L2 loss, with client c_1 (c_2) minimizing $L_1 = \|f(x_1) - y_1\|^2$ ($L_2 = \|f(x_2) - y_2\|^2$), where f denotes the model.

Model Architecture. The model architecture is a two-layer linear network f with two nodes $[a, b]$ in the first layer and one node $[v]$ in the second layer. Specifically, $f(x) = x[a, b]^\top c$. The task setup implies that client c_1 (c_2) aims to minimize $L_1 = |av - 1|^2$ ($L_2 = |bv - 1|^2$).

The chosen setup, which includes an orthogonal dataset distribution, regression tasks, and a two-layer neural network model, has been previously employed to study transfer learning with out-of-distribution datasets [19] and domain generalization [21]. Specifically, [19] demonstrates that fine-tuning on out-of-distribution data can lead to feature distortion on in-distribution data. Meanwhile, [21] focused on the case of fine-tuning only the pre-trained encoder, which outperforms full-layer fine-tuning in domain generalization when the target data is insufficient. Our work contributes to the existing literature by extending previous approaches to a federated learning setup, where two clients have orthogonal dataset distributions. Addressing this new scenario introduces unique challenges and considerations that have not been explored in previous works.

B.2 Preliminary: FedAvg and FedBug

We review FedAvg and its notations, as illustrated in Figure 5. During the i -th global round, the server distributes parameters $[a^i, b^i, v^i]$ to the clients. For example, client c_1 receives the initial parameter $[a_{c_1,0}^i, b_{c_1,0}^i, v_{c_1,0}^i] (= [a^i, b^i, v^i])$. Each client individually optimizes cost function using their own parameters. After k local iterations, the parameters of client c_1 are updated to $[a_{c_1,k}^i, b_{c_1,k}^i, v_{c_1,k}^i]$, and upon achieving local convergence, they become $[a_{c_1,*}^i, b_{c_1,*}^i, v_{c_1,*}^i]$. Once local convergence is reached, clients send their learned parameters back to the server. The server then averages

the received parameters to obtain the new parameters $[a^{i+1}, b^{i+1}, v^{i+1}] = \frac{1}{2}([a_{c_1,*}^i, b_{c_1,*}^i, v_{c_1,*}^i] + [a_{c_2,*}^i, b_{c_2,*}^i, v_{c_2,*}^i])$. We denote the solution obtained at global convergence as $[a^*, b^*, v^*]$. Notably, a solution must satisfy $a^* = b^*$ since $a^*v^* = 1$ and $b^*v^* = 1$.

The proposed FedBug differs from FedAvg only in the client-side update step: Clients first freeze v and update $[a, b]$ for a few local iterations. Afterwards, the client unfreezes the last layer parameters and performs gradient descent on all the parameters.

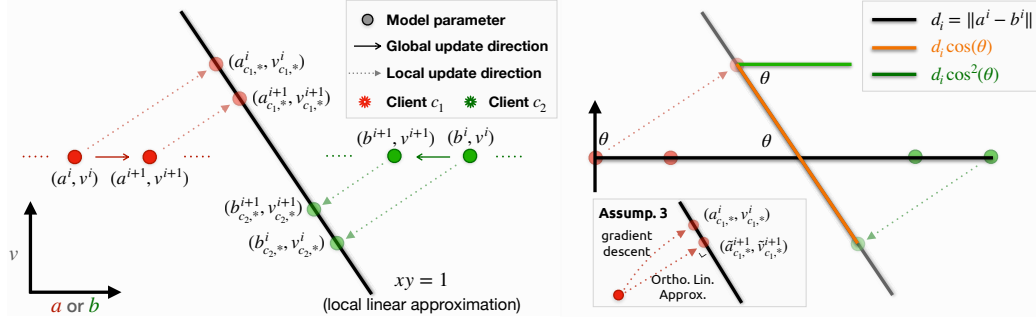


Figure 5: **Left: Notation for our theoretical analysis.** The horizontal axis concurrently signifies the parameter axis for parameters a and b . This approach enables us to visually explain the concept of client discrepancy, a term we define and utilize later. **Right: Visualization for Theorem 3 and Assumption 6.** Different colors indicate the segments needed to calculate the client discrepancy contraction ratio. A higher alignment between the local update direction (represented by the dotted line) and the v axis results in a larger contraction ratio. The notion behind Assumption 6 is also visually illustrated.

B.3 The Learning Dynamics of Models Trained With FedAvg

In this subsection, we provide an overview of the learning dynamics of models trained using the FedAvg algorithm. Specifically, at the client-side, if models take an infinitely small learning rate such that the optimization process becomes a gradient flow [4, 5, 7, 44], the parameters' learning trajectory follows a hyperbolic track. Thus, each client's minimizers can be analytically derived due to the deterministic nature of the trajectory. At the server-side, the minimizers of the clients are averaged, leading to a discontinuity between each client's minimizers and the averaged parameters, presenting a challenge in discussing the continuous learning behavior of the model. To overcome this challenge, we introduce assumptions that enable us to better analyze the learning behavior.

Local Iteration (Client Side). In this section, we derive the local learning dynamics of the parameters for an infinitely small step size. We consider a gradient flow, implying that the model parameters are functions of time t . We denote the parameters of the model for client c_1 at time t as $a(t)$, $b(t)$, and $v(t)$. A similar derivation applies for client c_2 .

The gradient of the model at time t can be defined as follows:

$$\begin{aligned} \frac{da(t)}{dt} &= -\eta \frac{\partial L_1}{\partial a(t)} = -\eta v(t)(a(t)v(t) - 1), \\ \frac{db(t)}{dt} &= -\eta \frac{\partial L_1}{\partial b(t)} = 0, \\ \frac{dv(t)}{dt} &= -\eta \frac{\partial L_1}{\partial v(t)} = -\eta a(t)(a(t)v(t) - 1). \end{aligned}$$

Subsequently, by multiplying $\frac{da(t)}{dt}$ by $a(t)$ and $\frac{dv(t)}{dt}$ by $v(t)$, we obtain:

$$a(t) \frac{da(t)}{dt} = v(t) \frac{dv(t)}{dt}.$$

This equality leads to the following conservation law:

$$a(t)^2 - v(t)^2 = a(0)^2 - v(0)^2 = a^i{}^2 - v^i{}^2.$$

Assuming that $a^i{}^2 - v^i{}^2$ is not zero, this equation represents a hyperbola in the (a, v) plane, this suggests that the evolution of the parameters $a(t)$ and $v(t)$ follows a hyperbolic trajectory.

The local minimizers can be derived as follows. Recall that the learning trajectory at the client side is hyperbolic and the final solution must satisfy $av = 1$. Using high school algebra, we obtain the following analytic solutions:

$$\begin{aligned} a_{c_1, \infty}^i &= a(\infty) = \frac{1}{2} \sqrt{\frac{1}{2} (\sqrt{(a^i{}^2 - v^i{}^2)^2 + 4} + (a^i{}^2 - v^i{}^2))}, \\ v_{c_1, \infty}^i &= v(\infty) = \frac{1}{2} \sqrt{\frac{1}{2} (\sqrt{(a^i{}^2 - v^i{}^2)^2 + 4} - (a^i{}^2 - v^i{}^2))}. \end{aligned}$$

Global Round (Server Side). This section examines the behavior of the averaged client parameters. According to the update rules of FedAvg, we have the following:

$$\begin{aligned} a^{i+1} &= \frac{1}{2} (a_{c_1, \infty}^i + a^i), \\ b^{i+1} &= \frac{1}{2} (b^i + b_{c_2, \infty}^i), \\ v^{i+1} &= \frac{1}{2} (v_{c_1, \infty}^i + v_{c_2, \infty}^i) \\ &= \frac{1}{4} \left(\sqrt{\frac{1}{2} (\sqrt{(a^i{}^2 - v^i{}^2)^2 + 4} - (a^i{}^2 - v^i{}^2))} \right. \\ &\quad \left. + \sqrt{\frac{1}{2} (\sqrt{(b^i{}^2 - v^i{}^2)^2 + 4} - (b^i{}^2 - v^i{}^2))} \right) \end{aligned}$$

It is important to note that during local training, client c_1 only updates parameters a and v , leaving b unchanged. Therefore, $b_{c_1, \infty}^i = b_{c_1, 0}^i = b^i$. The same principle applies to client c_2 with respect to its own parameters.

B.4 Challenges in Analysis

To the best of our knowledge, we are the first to explore the learning dynamics and convergence behavior of FL algorithms in an over-parameterized model and orthogonal dataset setting. This setup presents challenges not previously addressed in FL analysis. (1) **Non-convexity**: The model is over-parameterized, which leads to a non-convex loss landscape, making optimization challenging without additional constraints or restrictions. To illustrate the non-convex of the loss function, we can evaluate the second derivative of client c_1 's loss function $L_1 = (av - 1)^2$ with respect to the parameters a and v . By computing the Hessian matrix and its eigenvalues, we can get insight about the convexity of the function. For instance, at the parameter values $a = 0.1$ and $v = 0.1$, the Hessian matrix yields one negative eigenvalue, indicating the non-convexity. (2) **Hyperbolic local trajectories**: As shown in the previous subsection, the local trajectories of client-side parameters can follow a hyperbolic path. This adds complexity to the analysis of client-side learning dynamics, as the intersection point with the global minimum may not exhibit desirable properties and local convexity may not be clear.

These challenges make the analysis of FedAvg in an over-parameterized model and orthogonal dataset setting a complex problem. To facilitate theoretical analysis and provide desirable guarantees, our assumptions are necessary.

B.5 The Convergence Rate of FedAvg and FedBug

In this section, we prove that FedBug converges faster than FedAvg. We begin with the introduction of the three assumptions and then proceed with our two theorems.

Assumption 4. (Local Linearity) *The server model parameters are initialized in the vicinity of the global minimum, such that the global minimum can be locally approximated as a linear function.*

Assumption 5. (Bounded Local Convexity) Under Assumption 4, there exist constants $\beta_1 > 0$ and $0 < \beta_2 < 1$, such that for all n , the value of v^{n^2} is bounded as follows: $\beta_1\beta_2 \leq v^{n^2} \leq \beta_1$.

Assumption 6. (Orthogonal Trajectory and Bounded Error) Under Assumption 4 and Assumption 5, with a properly chosen step size, the local gradient descent update trajectory can be approximated by a linear trajectory orthogonal to the global minimum. The approximation error, which quantifies the discrepancy between the gradient descent solution and the ideal orthogonal linear path solution, is bounded by α times the length of the orthogonal linear trajectory.

Assumption 4 ensures that FedAvg and FedBug starts close enough to the global minimum, enabling convergence without the interference of poor initialization and unbounded convexity. Building upon Assumptions 4 and 5 further characterizes the local convexity of the objective function around the global minimum. By ensuring that it is bounded, both FedAvg and FedBug are prevented from diverging or oscillating around the global minimum when using an appropriate step size. Lastly, Assumption 6 guarantees that FedBug converges along a favorable direction and help us to quantify the improvement brought by single global round, while allowing for quantification of the approximation error. Please refer to the right panel of Figure 5 for illustration. We defer the discussion of limitation to Section B.7.

In this section, we present our theoretical findings on the convergence behavior of FedAvg. Distinct from conventional FL convergence analysis, we discover that the unique FL setup permits an alternative approach for proving convergence. The cornerstone of our theorem is the observation that at each global round, the discrepancy between clients' parameters diminishes by a factor r . Here, r is determined by the degree of alignment between the local update and the axial direction of the last layer parameters – a measure that captures the consistency of local updates with the global model's structure.

To measure this discrepancy reduction ratio, we define two useful terms:

Definition 3. Client discrepancy d^i is the L1 distance between the server model parameters a^i and b^i at the i -th global round: $d^i = \|a^i - b^i\|$.

Definition 4. Client discrepancy contraction ratio r : $r = d^{i+1}/d^i$.

Now, we present the theorem describing the convergence behavior of FedAvg:

Theorem 3. Under Assumptions 4, 5 and 6, models trained using FedAvg converge with the client discrepancy contraction ratio upper bounded by $\frac{1+\cos^2\theta(1+\alpha)}{2}$, where θ is the angle between the local update direction and the axis of the last layer parameter.

Proof. The proof aims to show that the client discrepancy contracts after one global round. The contraction ratio is determined by the angle between the axes of last layer parameters and the local update direction (refer to Figure 5 for illustration).

For conciseness, we denote the minima client c_1 and c_2 reached under Assumption 6 as $(\tilde{a}_{c_1,*}^i, \tilde{b}_{c_1,*}^i)$ and $(\tilde{a}_{c_2,*}^i, \tilde{b}_{c_2,*}^i)$, respectively. We can express the client discrepancy at the i -th global round as follows:

$$d^{i+1} = \|a^{i+1} - b^{i+1}\| = \left\| \frac{a^i + a_{c_1,*}^i}{2} - \frac{b^i + b_{c_2,*}^i}{2} \right\| = \frac{1}{2}d^i + \frac{1}{2}\|a_{c_1,*}^i - b_{c_2,*}^i\| \quad (1)$$

Considering the approximation error stated in Assumption 6, we get:

$$\frac{1}{2}\|a_{c_1,*}^i - b_{c_2,*}^i\| \leq \|\tilde{a}_{c_1,*}^i - \tilde{b}_{c_2,*}^i\| + e_i, \quad (2)$$

where $0 < e_i \leq \alpha\|\tilde{a}_{c_1,*}^i - \tilde{b}_{c_2,*}^i\|$ is the approximation error.

To obtain $\|\tilde{a}_{c_1,*}^i - \tilde{b}_{c_2,*}^i\|$, we notice that it is the projection of the line segment connecting $(\tilde{a}_{c_1,*}^i, \tilde{b}_{c_1,*}^i)$ and $(\tilde{a}_{c_2,*}^i, \tilde{b}_{c_2,*}^i)$ onto the plane orthogonal to the axial direction of v . The line segment, on the other hands, represent the projection of the original client discrepancy on to the linearized global minima. Therefore, $\|\tilde{a}_{c_1,*}^i - \tilde{b}_{c_2,*}^i\| = d^i \cos^2 \theta$, as it is a double projection of the original client discrepancy.

Substituting this into the expression for d^{i+1} , we get:

$$d^{i+1} = \frac{1}{2}d^i + \frac{1}{2}\|a_{c_1,*}^i - b_{c_2,*}^i\| \leq \frac{1}{2}d^i + \frac{1}{2}d^i \cos^2 \theta(1 + \alpha) \quad (3)$$

This yields the ratio

$$r = \frac{d^{i+1}}{d^i} \leq \frac{1 + \cos^2 \theta (1 + \alpha)}{2} \quad (4)$$

□

We proceed to theoretically demonstrate the superiority of FedBug over FedAvg by proving that FedBug exhibits a smaller contraction ratio than FedAvg. To better comprehend the learning behavior, we focus on the case that the last layer parameter c is frozen only for *one local training iteration* and simultaneously perform gradient descent only on parameters a and b . Afterwards, all clients unfreeze parameter v and conduct gradient descent on all parameters. Importantly, unlike the learning dynamics of FedAvg, we consider a finite step size in this analysis.

Theorem 4. *Under Assumptions 4, 5, and 6, there exists $1 - \beta_2 < m < 1$ such that if the step size satisfies $\frac{1-m}{\beta_1\beta_2} < \eta < \frac{1}{\beta_1}$, FedBug converges with a client discrepancy contraction ratio upper bounded by $\frac{1+m\cos^2\theta(1+\alpha)}{2}$, where θ is defined as in Theorem 3.*

Proof. To prove Theorem 4, we divide the learning process into two stages. In the first stage, we update the parameters $a_{c1,0}^i$ and $b_{c2,0}^i$ for one step while freezing the last layer parameter c . Therefore we have:

$$\begin{aligned} a_{c1,1}^i &= a^i - \eta v^i (a^i v^i - 1), \\ b_{c2,1}^i &= b^i - \eta v^i (b^i v^i - 1). \end{aligned}$$

We now compute the distance between the updated parameters:

$$\|a_{c1,1}^i - b_{c2,1}^i\| = \|(a^i - b^i)(1 - \eta v^{i2})\|. \quad (5)$$

In the second stage, we unfreeze parameter v and perform gradient descent on all the parameters. Then, we can obtain $\|\tilde{a}_{c1,*}^i - \tilde{b}_{c2,*}^i\| = \|a_{c1,1}^i - b_{c2,1}^i\| \cos^2 \theta$ similar to Theorem 3 and get:

$$\begin{aligned} d^{i+1} &\leq \frac{1}{2}d^i + \frac{1}{2}\|\tilde{a}_{c1,*}^i - \tilde{b}_{c2,*}^i\|(1 + \alpha) \\ &= \frac{1}{2}d^i + \frac{1}{2}\|a_{c1,1}^i - b_{c2,1}^i\| \cos^2 \theta (1 + \alpha) \\ &= \frac{1}{2}d^i + \frac{1}{2}(1 - \eta c^{i2})\|a_{c1,0}^i - b_{c2,0}^i\| \cos^2 \theta (1 + \alpha) \end{aligned} \quad (6)$$

Using Assumption 5 and the condition on step size, we obtain the contraction ratio:

$$r = \frac{1 + (1 - \eta c^{i2}) \cos^2 \theta (1 + \alpha)}{2} < \frac{1 + m \cos^2 \theta (1 + \alpha)}{2} \quad (7)$$

□

Implication From the Proof. Our proof introduces a novel approach to analyzing the convergence behavior of the FedAvg algorithm in an over-parameterized context, contributing a significant insight to the FL community. We extend beyond the traditional single-variable loss function analysis used in previous studies, tackling a more complex multi-variable loss scenario. At the same time, our unique orthogonal task setup also practically captures the client drift caused by dataset heterogeneity. Moreover, we propose a novel measure of client alignment - the contraction ratio r , which quantifies the alignment between local updates and the global model's structure, thereby offering a more intuitive understanding of convergence dynamics. Our findings point to a promising future research direction - optimizing local-global alignment to enhance convergence in over-parameterized models and orthogonal regression datasets. Our work not only deepens the understanding of FL algorithms but also paves the way for creating more efficient FL systems.

B.6 Empirical Validation of Convergence Rate

Experimental Setup. We conduct simulation experiments to empirically verify our theorem and the underlying assumptions. In our experiments, we consider a two-client federated learning setup with a two-layer network, following the orthogonal dataset setup described in the theoretical analysis. The model parameters (a^0, b^0, v^0) are initialized uniformly from the interval $[0, 2]$, and we repeat the experiments 50 times with different seeds. For experimental details, we perform 80 global rounds in each experiment. On the local side, we utilize the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.1, and each client model is trained for 50 local iterations. To compute the theoretical contraction ratio near the minima, we explicitly estimate θ using the gradient of the client model and then compare it with the empirical one $r = \frac{d^{i+1}}{d^i}$, where d^i represents the difference between a^i and b^i .

B.7 Discussions

Relation to FedBABU. Our analysis extends to FedBABU, in which the last layer parameter v remains constant throughout the training. In this context, the client discrepancy contraction ratio reduces to $\frac{1}{2}$. Despite the rapid convergence of FedBABU, it encounters a significant challenge. Similar to Proposition 2 presented in [21], we can consider a scenario where a non-linear function (e.g., Rectified Linear Unit) is positioned between the first and second layers and v is initialized with a negative value. In such a situation, FedBABU falls short of achieving the optimum because the model output always remains non-positive, while both FedAvg and FedBug can still converge to some stationary points.

Discussion About Assumptions. Our analysis is based on three key assumptions that help us derive tractable convergence results, thereby simplifying the complex task of dealing with the global parameter sequence.

- The first assumption (Local Linearity) considers the global minimum as a linear function, allowing for the direct quantification of the client discrepancy and the double projection trick, making the derivation more straightforward. However, we recognize that this approximation introduces a margin of error that could influence the overall model performance and the applicability of the theoretical results. Future work may aim to quantify this error consider first-order or second-order Taylor expansion and determine an upper bound to better understand its impact.
- Assumption 2 (Bounded Local Convexity) aligns with previous work in the Federated Learning (FL) domain. It bounds the local convexity, preventing the algorithms (FedAvg and FedBug) from diverging or oscillating when using an appropriate step size.
- Assumption 3 (Orthogonal Trajectory) builds upon Assumption 1 and explicitly bounds the approximation error to account for potential curvature in the gradient flow. This error is most pronounced at the beginning of the parameter update and could influence the algorithms' convergence trajectory. Interestingly, as the parameters get closer to the linearized global minima, their direction becomes increasingly orthogonal to the global minima, a consequence of the first assumption. Future research should explore how this error evolves during training and how it impacts the final model performance.

Future Directions. We briefly explore possible generalization of our presented work as the future direction from the task and model perspective. **Task.** First of all, we provide a direct generalization of the orthogonal task setup, where we use $\mathcal{T}_1 = \{x_1 = e_1, y_1 = 1\}$ and $\mathcal{T}_2 = \{x_2 = e_2, y_2 = 1\}$ as datasets. Here, e_1 and e_2 are orthogonal vectors, and we can define the model function as $f(x) = x[ae_1, be_2]^\top v$. Besides, we can also consider the case of m clients and redefine the client discrepancy to accommodate such scenarios. Moreover, the effect of dataset orthogonality and non-orthogonality can be studied to understand the effect of the general convergence of FL algorithms. **Model.** Generalizing the model presents two interesting avenues for research. Firstly, we can consider feature dimensions greater than one, which adds complexity to the analysis due to increased over-parameterization. Secondly, extending the analysis to a multi-layer neural network opens up possibilities for observing phenomena such as acceleration, as previously explored in studies like [2, 36]. Investigating these directions will contribute to a more comprehensive understanding of the convergence behavior of FL algorithms.

C Comparison Between FedAvg and FedBug

We provide self-contained outlines of the FedAvg and FedBug algorithms in Algorithm 2 and Algorithm 3, respectively. The key difference, highlighted in red and blue, is that while FedAvg updates all M modules at each local iteration, FedBug unfreezes and updates one module at the beginning, progressively training an additional module every $\frac{PK}{M}$ local iterations. As FedBug does not require extra information like gradients, momentum, or regularization, it can be easily incorporated into other FL algorithms.

Algorithm 2 FedAvg

Notation:
 $\theta^{1:m}$: the first m modules of model θ
 R : number of global rounds
 K : number of local iterations

```

1: Input: global model  $\theta$  with  $M$  modules
2: for  $r = 1, \dots, R$  do
3:   Sample clients  $S \subseteq \{1, \dots, N\}$ 
4:   for each client  $i \in S$  in parallel do
5:     Initialize local model  $\theta_i \leftarrow \theta$ 
6:     for  $k = 1, \dots, K$  do
7:        $\theta_i^{1:M} \leftarrow \theta_i^{1:M} - \eta_l \nabla F_i(\theta_i^{1:M})$ 
8:     end for
9:      $\Delta_i \leftarrow \theta_i - \theta$ 
10:  end for
11:   $\theta \leftarrow \theta + \frac{\eta_g}{|S|} \sum_{i \in S} \Delta_i$ 
12: end for

```

Algorithm 3 FedBug

Notation:
 $\theta^{1:m}$: the first m modules of model θ
 R : number of global rounds
 K : number of local iterations
 P : gradual unfreezing stage percentage

```

1: Input: global model  $\theta$  with  $M$  modules
2: for  $r = 1, \dots, R$  do
3:   Sample clients  $S \subseteq \{1, \dots, N\}$ 
4:   for each client  $i \in S$  in parallel do
5:     Initialize local model  $\theta_i \leftarrow \theta$ 
6:     for  $k = 1, \dots, K$  do
7:        $m \leftarrow \min\{M, \lceil \frac{kM}{PK} \rceil\}$ 
8:        $\theta_i^{1:m} \leftarrow \theta_i^{1:m} - \eta_l \nabla F_i(\theta_i^{1:m})$ 
9:     end for
10:     $\Delta_i \leftarrow \theta_i - \theta$ 
11:  end for
12:   $\theta \leftarrow \theta + \frac{\eta_g}{|S|} \sum_{i \in S} \Delta_i$ 
13: end for

```
