

857	Contents	
858	1 Introduction	1
859	2 Related Work	3
860	3 Empirical Study of PFT	3
861	3.1 Overview and Definitions	3
862	3.2 Empirical Analysis Settings	4
863	3.3 Global and Local Performance Trends in PFT Baselines	5
864	3.4 Performance Comparison	5
865	3.5 Insight and Explanation on the Observations	7
866	4 Theoretical Analysis of the LP-FT in FL	7
867	4.1 LP-FT's Global Performance Under Concept Shift	8
868	4.2 LP-FT's Global Performance under Combined Concept and Covariate Shifts	9
869	5 Conclusion	9
870	A Related Work: Personalized Federated Learning and Fine-Tuning	22
871	B Different Variants of Fine-tune Strategy	23
872	B.1 Proximal FT	23
873	B.2 Soup FT	23
874	B.3 Sparse FT	23
875	B.4 LSS FT	24
876	B.5 LP-FT	24
877	C Experimental Details	24
878	C.1 Computing Environment and Hyper-parameters	24
879	C.2 Visualization of Original Images	24
880	C.3 Detailed Dataset and Model Information	26
881	D Further Empirical Results	27
882	D.1 Other PEFT Methods Distort Federated Feature	27
883	D.2 Label Shift	27
884	D.3 Further Empirical Validations for Theoretical Findings	28
885	E Proofs	28
886	F Empirical Performance of LP-FT and FT Under Theorem 4.5 Conditions	37
887	G Computational Overhead of LP-FT Relative to FT	38

Roadmap of Appendix. This appendix provides all supplemental material referred to in the main paper. First, in Section A we review related work on personalized federated learning (FL) and fine-tuning. Section B introduces the variants of fine-tuning strategies evaluated in our experiments, including Proximal FT, Soup FT, Sparse FT, LSS FT, and LP-FT. Section C gives full experimental details: computing environment, hyperparameters, dataset visualizations (Fig. 5), and dataset/model statistics (Table 4). In Section D we present additional empirical results, including PEFT distortions (Table 5) and label-shift experiments (Table 6). Section B.1 contains full proofs of Lemma 4.3, Theorems 4.4 and 4.5, along with illustrative plots (Fig. 6). Finally, Section G analyzes the computational overhead of LP-FT vs. FT.

Table 3: Important notations used in the appendix

Notation	Description
C	Number of clients in federated learning
m	Dimension of the common part of the linear head shared across clients
d	Dimension of feature representations
k	Dimension of the linear head
$B \in \mathbb{R}^{k \times d}$	Feature extractor matrix
$B_* \in \mathbb{R}^{k \times d}$	Ground-truth feature extractor matrix (with orthonormal rows)
$V \in \mathbb{R}^k$	Linear head weight vector
$V_i^* \in \mathbb{R}^k$	Ground-truth linear head for client i
$V_0 \in \mathbb{R}^k$	Initial linear head before fine-tuning
$V_{\text{com}}^* \in \mathbb{R}^m$	Common component of linear heads across clients
λ	Concept shift magnitude (heterogeneity parameter)
ϵ	Noise scale in the covariate shift model
\mathcal{D}_i	Data distribution of client i
\mathcal{D}_G	Mixture of all clients' data distributions (global distribution)
η	Learning rate used during fine-tuning
θ_L	Parameters of the local model
θ_G	Parameters of the global model
\mathcal{L}_L	Local loss function
$\hat{\mathcal{L}}_L$	Empirical local loss
\mathcal{L}_G	Global loss function

A Related Work: Personalized Federated Learning and Fine-Tuning

Personalized FL. Heterogeneous FL refers to a decentralized training paradigm that accommodates diverse and disparate data sources or devices participating in a collaborative model-building process. Examples include FedAvg [37] and various improvements in terms of aggregation optimization and local optimization. It is shown to experience challenges in heterogeneous scenarios. Thus, various literature proposes alternative strategies. Here, we summarize these strategies into aggregation optimization and local optimization. FedNova [45] belongs to the category of Aggregation optimization, which normalizes and scales the local updates. Examples of local optimization include FedProx [32] and Scaffold [25], where FedProx adds a L_2 regularization for each client and Scaffold adds a variance reduction term. However, these methods often exhibit limited personalization capabilities and may not adequately meet the performance requirements of different clients. Consequently, various personalized FL approaches have been proposed, with a primary emphasis on enhancing local client performance to the greatest extent possible. We can group these personalized FL strategies into clustering-based methods [10], transfer learning [50], and interpolating the local and global models [36, 8]. Some FL methods [11, 41] highlight that existing federated learning methods often fail under feature shift despite addressing label shift, proposing clustering and regularization strategies respectively to tackle diverse distribution shifts in non-IID data settings.

Fine-Tuning. Fine-tuning pre-trained models has become increasingly popular with the rise of foundation models [1]. However, fine-tuning with limited data often lead to overfitting. Several strategies can mitigate this issue, such as using optimizers that promote a flatter loss landscape [30, 23]. Notably, Sharpness-Aware Minimization (SAM) [9] and Stochastic Weight Averaging (SWA) [21]

are two popular methods that help achieve this. Additionally, a technique called *model soups* [47], uses a simple greedy weight averaging approach similar to SWA, shown significant improvements in fine-tuning. An interesting perspective focuses on minimizing the linear mode connectivity barrier between the pre-trained and fine-tuned models, helping maintain consistency in decision-making mechanisms from a loss landscape perspective [44]. *Partial fine-tuning* is another common method to prevent overfitting, which involves selectively fine-tuning specific layers of the model to better adapt to variations in data distribution [29]. Recent studies have introduced the concept of LP-FT [27], highlighting potential distortions in pre-trained features and their underperformance in scenarios involving previously unseen data. Further research on LP-FT provides a deeper analysis of model adaptation [43], focusing on feature distortion and simplicity bias, thereby enhancing our understanding of fine-tuning mechanisms and safe model adaptation.

B Different Variants of Fine-tune Strategy

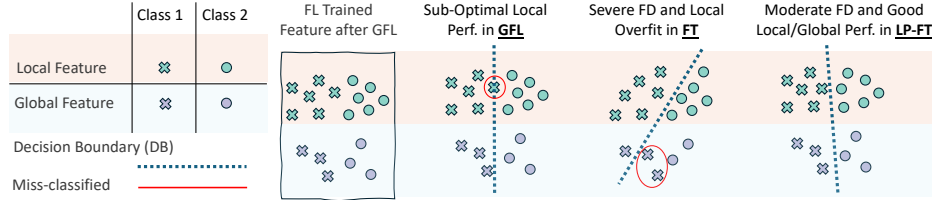


Figure 4: Illustration of federated feature distortion (FD) and decision boundaries.

This section provides an overview of the baseline techniques utilized in our study. We describe the characteristics and implementation specifics of three main fine-tuning methods: Proximal FT, Soup FT, and Sparse FT.

B.1 Proximal FT

Proximal Fine-Tuning (Proximal FT) [31] is a method that emphasizes preserving the original knowledge of the pre-trained model while adapting it to new tasks. This technique employs proximal regularization, which penalizes large deviations from the initial model parameters during the fine-tuning process. The primary advantage of Proximal FT is its ability to maintain the generalization capabilities of the pre-trained model, thus reducing the risk of overfitting to the new task’s data. In our experiments, we used an L2 regularization term to enforce proximity between the pre-trained and fine-tuned weights, with a regularization coefficient of 0.01.

B.2 Soup FT

Soup Fine-Tuning (Soup FT) [46] is an innovative approach that leverages the concept of "model soups," where multiple fine-tuned models are combined to create a more robust final model. The key idea is to fine-tune several instances of the pre-trained model on the target task with different random initializations or data shuffling, and then average the resulting weights to form a "soup." This method aims to enhance model robustness and performance by integrating the strengths of various fine-tuning instances. For our implementation, we fine-tuned five versions of the pre-trained model and averaged their parameters to create the final Soup FT model.

B.3 Sparse FT

Sparse Fine-Tuning (Sparse FT) [28] introduces sparsity constraints into the fine-tuning process, encouraging the model to update only a subset of its parameters. This approach aims to improve model efficiency and interpretability by ensuring that only the most relevant weights are adjusted during training. Sparse FT can be particularly beneficial for deploying models in resource-constrained environments where computational efficiency is paramount. In our experiments, we applied a gradient-based metric to the parameter prune, setting the regularization coefficient to 0.001 to achieve a balance between performance and sparsity.

B.4 LSS FT

In the context of FL, the Local Superior Soups (LSS) methodology [3] introduces a novel approach to enhance both generalization and communication efficiency. By leveraging model interpolation-based local training, LSS encourages clients to explore a connected low-loss basin through optimizable and regularized model interpolation. This strategy not only mitigates the challenges posed by data heterogeneity but also significantly reduces the number of communication rounds required for model convergence. Empirical evaluations have demonstrated the effectiveness of LSS across various widely used FL datasets, underscoring its potential as a catalyst for the seamless adaptation of pre-trained models in federated settings. In our experiments, we fine-tuned three candidate models for model interpolation and averaged their parameters to create the final Soup FT model.

B.5 LP-FT

Linear Probing and then Fine-Tuning (LP-FT) [27] is a two-step transfer learning approach designed to balance in-distribution (ID) and out-of-distribution (OOD) performance. In the first step, linear probing trains only the final layer (head) while freezing the pretrained feature extractor to ensure OOD robustness. The second step fine-tunes all model parameters to improve ID accuracy while retaining the benefits of linear probing for OOD generalization. LP-FT addresses the trade-offs inherent in full fine-tuning by initializing with a well-aligned linear head, reducing feature distortion during optimization. Empirically, LP-FT demonstrates superior ID and OOD accuracy across diverse datasets.

C Experimental Details

C.1 Computing Environment and Hyper-parameters

All experiments in this paper are conducted on NVIDIA A40 Graphics cards using PyTorch. The Adam optimizer is employed with a learning rate of 1×10^{-3} . In FL for all datasets, the standard local model update epochs are set to 1. The communication round is set to be 100 epochs, where we validated the model results from FL converged. Unless specified otherwise, the batch size for all benchmarks is standardized at 128. To ensure a fair comparison with various baselines, all methods initiate the FL personalized fine-tuning with models derived from the best-performing global model in terms of overall effectiveness.

C.2 Visualization of Original Images

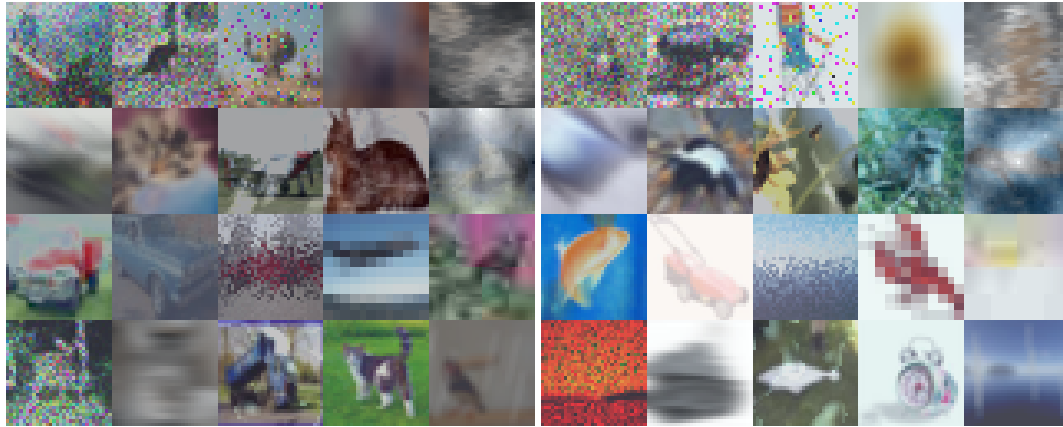
Datasets and Distribution Shifts. Figure 5 gives an overview of the benchmarks used in this study, organized by the four types of distribution shift we investigate—*feature-*, *input-*, *output-*, and *label-level*. All datasets are chosen to mirror the heterogeneity observed in real-world federated learning (FL) deployments.

- **Feature-level shift: Digit5 and DomainNet.** **Digit5** merges five digit domains—MNIST, SVHN, USPS, SynthDigits, and MNIST-M—whose diverse styles (e.g. grayscale scans vs. synthetic colors) induce substantial feature variations. **DomainNet** depicts everyday objects across six artistic domains (clip art, sketch, photo, *etc.*), further stressing the model’s ability to transfer features across drastically different visual styles.
- **Input-level shift: CIFAR10-C and CIFAR100-C.** CIFAR10-C/100-C apply 50 corruption types (Gaussian noise, motion blur, pixelation, brightness, contrast, . . .) to the original CIFAR images, emulating degradations caused by hardware or environment. These pixel-level perturbations leave semantics intact while challenging a model’s robustness to distorted inputs—crucial for FL settings such as mobile sensing and autonomous driving.
- **Output-level shift: CheXpert and CelebA.** In **CheXpert**, we focus on two labels (Edema, No Finding) and partition clients to introduce demographic biases (e.g. male patients predominating in Edema, female in No Finding). **CelebA** clients are built by correlating gender and hair-color attributes (e.g. “blonde-haired females” vs. “non-blonde males”), yielding skewed label distributions that test a model’s resilience to demographic imbalance.

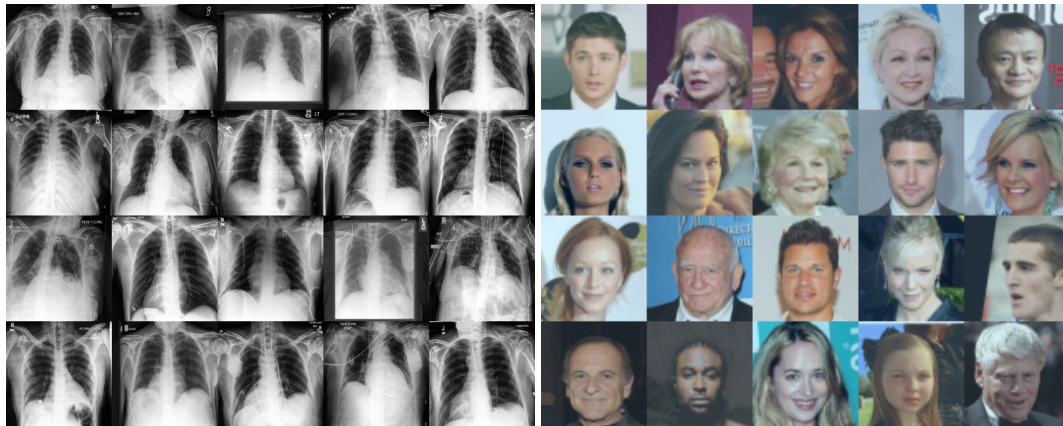
- **Label-level shift: CIFAR10 (Dirichlet- $\alpha=0.1$).** We partition the clean CIFAR-10 dataset among 20 clients using a Dirichlet prior with $\alpha = 0.1$, producing highly imbalanced class proportions. Such label shift typifies FL scenarios where each client serves a niche population—e.g. hospitals specializing in different diseases—violating the IID assumption of standard aggregation rules.



(a) Feature-level Shift (Digit5 and DomainNet)



(b) Input-level Shift (CIFAR10-C and CIFAR100-C)



(c) Output-level Shift (ChexPert and CelebA)

Figure 5: Visualization of the original datasets used in the paper.

C.3 Detailed Dataset and Model Information

Table 4 provides a visual overview of the datasets used in this study, categorized by their levels of transformation and data heterogeneity. The table is divided into four sections, corresponding to feature-level, input-level, output-level, and label shift settings:

Feature-Level Shift (Digit5 and DomainNet): The **Digit5** dataset, which consists of digit images collected from five distinct domains, including MNIST, SVHN, USPS, SynthDigits, and MNIST-M. These images exhibit a variety of styles, such as handwritten digits, digits rendered in different fonts, and textured representations, demonstrating substantial visual heterogeneity. The **DomainNet** dataset is a large-scale collection featuring objects and scenes from six domains, including styles like clip art, sketches, and realistic photographs.

Input-Level Shift (CIFAR10-C and CIFAR100-C): This type of distribution shift includes corrupted versions of CIFAR-10 and CIFAR-100 datasets. The **CIFAR10-C** dataset applies 50 types of corruptions, such as noise, blur, and distortions, to evaluate model robustness under various degradation conditions. Similarly, **CIFAR100-C** extends the CIFAR-100 dataset by introducing the same set of corruptions, enabling robustness evaluation on a larger and more diverse set of categories.

Data Heterogeneity Type	Dataset	Model	Description	Clients	Classes
Feature-Level Shift	Digit5	ResNet18	A collection of digit images from five domains, used for domain adaptation and digit recognition tasks [16]. Datasets include MNIST, SVHN, USPS, SynthDigits, and MNIST-M.	5	10
	DomainNet	ResNet50	A large-scale dataset of images from six distinct domains for multi-source domain adaptation [40]. Preprocessing follows the strategy in FedBN [34].	6	10
Input-Level Shift	CIFAR10-C	ResNet18	A corruption benchmark dataset for CIFAR-10 [13], augmented with 30 additional corruption types [38] and one extra type [4], totaling 50 corruption types (including the original 19 types of corruption from CIFAR-10-C, plus 30 additional corruption types and one extra type, resulting in a total of 50 distinct corruption types).	50	10
	CIFAR100-C	ResNet18	An extension of CIFAR-100 with common corruptions, following the same strategy as CIFAR-10-C.	50	100
Output-Level Shift	CheXpert	ResNet50	A chest radiograph dataset labeled for 14 common chest conditions [18]. Edema and No Finding labels are grouped as described in [22]. Clients are spuriously correlated with the attribute gender (e.g., 90% of label 1 examples in a client are male).	2	2
	CelebA	ResNet50	Over 200,000 celebrity images with 40 attributes [35]. Client splitting follows the same strategy as CheXpert, with attributes: male, female, blonde hair, and non-blond hair.	4	2
Label Shift	CIFAR10	ResNet18	A benchmark dataset with label shift induced via Dirichlet distribution ($\alpha = 0.1$), distributed across 20 clients [26].	20	10

Table 4: Detailed information about the datasets used in the study. For all the settings above, each client has an individual data distribution, while different clients prohibits the corresponding shifts to ensure the non-IID nature required for heterogeneous FL. Feature-level shift, also referred to as subgroup shift, and input-level shift, corresponding to image corruption, are categorized as covariate shift. Output-level shift, representing spurious correlations in our setting, is categorized as concept shift.

Output-Level Shift (CheXpert and CelebA): The **CheXpert** dataset, a widely used medical imaging dataset labeled for 14 common chest conditions. In this study, the Edema and No Finding labels are grouped, and spurious correlations are introduced at the client level, where attributes (i.e. gender in our client splitting) are disproportionately represented (i.e., 90% of label 1 examples in a client are a certain attribute). The **CelebA** dataset, which includes over 200,000 celebrity faces annotated with 40 attributes. Client splitting follows the same strategy as CheXpert, with attributes such as male, female, blonde hair, and non-blond hair used to create spurious correlations across clients.

Label Shift (CIFAR10): The final one focuses on label shift in the **CIFAR10** dataset. This setting simulates non-IID conditions by inducing label distributions across clients using a Dirichlet distribution with $\alpha = 0.1$. This creates significant variations in class distributions among the 20 clients,

1034 mimicking real-world federated learning scenarios where data availability across clients is inherently
1035 imbalanced.

1036 D Further Empirical Results

1037 D.1 Other PEFT Methods Distort Federated Feature

PEFT Method	Local	Global	Avg.
LoRA (lr=1e-3)	41.54	26.75	26.87
LoRA (lr=1e-4)	42.01	26.41	27.18
Adapter (lr=1e-3)	48.35	39.28	38.08
Adapter (lr=1e-4)	49.07	39.83	38.36
LP-FT	68.50	57.52	53.52

Table 5: Different PEFT compared on DomainNet with ViT.

1038 **Setup.** In this study, we adapted two widely recognized Parameter-Efficient Fine-Tuning (PEFT)
1039 methods: LoRA [15] and Adapter [14]. Both methods were fine-tuned with meticulous adjustments
1040 to their learning rates on the ViT, as detailed in Tab. 5. These configurations allowed us to assess the
1041 impact of different fine-tuning strategies on federated features. The performance of each method was
1042 measured in terms of local, global, and average accuracy.

1043 **Result.** The effectiveness of bias tuning in personalized fine-tuning naturally raises the question
1044 of whether other PEFT methods, commonly used for fine-tuning large models, exhibit similar
1045 effectiveness in our setting. In this study, we compare the local and global performance of other
1046 popular PEFT methods. Our findings reveal that while these methods can achieve high levels of
1047 local performance, their global performance still drops significantly, indicating that they distort the
1048 federated features to a certain extent. These PEFT methods’ local and global performance still fall
1049 short compared to LP-FT, indicating that they distort the federated features to a certain extent.

1050 The comparison of PEFT methods in the context of personalized fine-tuning sheds light on the unique
1051 challenges and requirements of this setting. Despite the success of PEFT techniques in fine-tuning
1052 large models for various tasks, our results suggest that their direct application to personalized FL
1053 may not yield optimal results in terms of preserving the global knowledge captured by the federated
1054 features.

1055 D.2 Label Shift

1056 We simulated label shift using a Dirichlet distribution with an alpha parameter set to 0.1. The dataset
1057 was distributed across 20 clients, and we trained a ResNet18 model for classification. Results is
1058 shown in Table 6.

	Baseline	Local	Global	C-Std.	Worst	Avg
CIFAR10	FT	87.20 (0.24)	16.67 (0.14)	26.81 (1.47)	0.01 (0.00)	34.62 (0.09)
	Proximal FT	89.80 (1.21)	17.42 (1.46)	27.31 (0.08)	0.01 (0.00)	35.75 (0.09)
	Soup FT	88.16 (0.15)	16.94 (0.05)	27.07 (0.11)	0.01 (0.00)	35.04 (0.04)
	Sparse FT	89.16 (0.00)	17.54 (0.14)	27.27 (0.02)	0.01 (0.00)	35.57 (0.04)
	LP-FT	90.15 (0.43)	17.73 (0.16)	27.37 (0.09)	0.01 (0.00)	35.96 (0.10)

Table 6: Comparison of PFT methods on CIFAR10 label shift setting.

1059 Tab. 6 compares different fine-tuning methods on CIFAR10 across multiple evaluation metrics,
1060 including local performance, global performance, robustness to corruption (C-Std.), worst-case
1061 performance, and average performance. Among the methods, LP-FT achieves the best results,
1062 excelling in local performance (90.15), global evaluation (17.73), and average performance (35.96).
1063 Proximal FT and Sparse FT also perform competitively, with improvements over standard fine-tuning

Table 7: Performance under label flipping for FT and LP-FT at different flipping ratios (F.R.).

Metric	F.R. 20%	F.R. 30%	F.R. 40%	F.R. 50%
FT Avg. \uparrow	67.73	60.04	58.27	60.06
LP-FT Avg. \uparrow	79.83	72.95	71.55	73.26
FT Global \uparrow	68.76	55.18	53.70	56.17
LP-FT Global \uparrow	83.08	72.75	69.89	72.32
FT Local \uparrow	91.32	91.12	90.84	91.88
LP-FT Local \uparrow	91.23	89.20	90.02	90.87

(FT) and Soup FT in most metrics. All methods show near-identical performance in the worst-case scenario (0.01), indicating a shared limitation in extreme cases. Overall, LP-FT demonstrates the most robust and effective fine-tuning approach on CIFAR10.

D.3 Further Empirical Validations for Theoretical Findings

Despite being based on simplified data and model assumptions, our theoretical results demonstrate significant practical relevance. In this section, we empirically validate the contributions in Sec. 4, exploring the performance implications of controllable heterogeneities in neural networks and datasets.

Experimental Settings. To validate the impact of λ in Theorem 4.5, we simulate a controllable concept shift setting on the Digit5 dataset with label-flipping under PFT for both FT and LP-FT. For each client, a proportion of labels is randomly flipped, referred to as the flipping ratio. For example, class one is flipped with class two for the first client, and class two with class three for the second, using a randomized mechanism. A higher flipping ratio indicates greater heterogeneity λ . The settings align with prior studies: the model is pre-trained within the FL framework and used to initialize both FT and LP-FT. This simulates the combined concept-covariate shift discussed in Sec. 4.2. Flipping labels reflects different labeling functions, where higher flipping rates indicate stronger concept shifts. The Digit5 dataset also introduces covariate shift, as outlined in Sec. 3.2.

Results. As shown in Tab. 7, LP-FT consistently outperforms FT in global performance across various flipping ratios. This aligns with our theoretical results in Sec. 4.2, especially for deep neural networks under realistic PFT settings. The flipping rate controls concept shift heterogeneity, with higher rates indicating greater heterogeneity, while varying data distributions introduce covariate shift. These experiments simulate the combined concept-covariate shift, as analyzed in our framework. Notably, LP-FT outperforms FT in all heterogeneity levels, validating its advantage in both low and high heterogeneity regimes (larger flipping ratios).

E Proofs

Proof of Lemma 4.3. We want to analyze the Fine-Tuning (FT) method, focusing on the effect of initial parameters. We perform one pass through the entire dataset to simulate the complete fine-tuning process. Consider the Mean Squared Error (MSE) loss function with parameters V and B , where B is represented as follows:

$$B = \begin{bmatrix} b_1^T \\ \vdots \\ b_m^T \\ b_{m+1}^T \\ \vdots \\ b_{m+C}^T \end{bmatrix},$$

where $b_i^T \in \mathbb{R}^{1 \times d}$ denotes the i -th row of matrix B , and $m + C = k$.

1094 To apply one step of gradient descent, we need to compute the gradient of the loss function with
 1095 respect to $V, b_1, b_2, \dots, b_{m+C}$, and then perform one update step.

1096 W.L.O.G. we assume the local client is client 1. We define the local loss function as follows:

$$\mathcal{L}_L(V, B) = \mathbb{E}_{x \sim \mathcal{D}_1} \left[\frac{1}{2} (V^T Bx - V_1^{*T} B_* x)^2 \right],$$

1097 where \mathcal{D}_1 is the data distribution for client 1.

1098 Now, let $(\mathbf{X}_1, \mathbf{Y}_1)$ represent the local dataset of client 1, consisting of n_1 data points $\{(x_{1j}, y_{1j})\}_{j=1}^{n_1}$.

1099 We aim to calculate the gradient of the empirical loss function with respect to the parameters. The
 1100 empirical loss function is given by:

$$\widehat{\mathcal{L}}_L(V, B) = \frac{1}{n_1} \sum_{j=1}^{n_1} \left[\frac{1}{2} (V^T Bx_{1j} - V_1^{*T} B_* x_{1j})^2 \right].$$

1101 In practice, we take the gradient of this empirical loss function with respect to the parameters $V, b_1,$
 1102 b_2, \dots, b_{m+C} . However, since we are particularly interested in computing the expectation $\mathbb{E}[b_j^{FT}]$,
 1103 we evaluate the expected value of the gradients using one pass through the whole dataset as follows:

$$\begin{aligned} \mathbb{E} \left[\left. \frac{\partial \widehat{\mathcal{L}}_L}{\partial V} \right|_{\substack{V=V_0 \\ B=B_*}} \right] &= \mathbb{E} \left[\left. \frac{\partial}{\partial V} \left(\frac{1}{n_1} \sum_{j=1}^{n_1} \frac{1}{2} (V^T Bx_{1j} - V_1^{*T} B_* x_{1j})^2 \right) \right|_{\substack{V=V_0 \\ B=B_*}} \right] \\ &= \frac{1}{n_1} \sum_{j=1}^{n_1} \mathbb{E} \left[(V_0^T B_* x_{1j} - V_1^{*T} B_* x_{1j}) x_{1j}^T B_*^T \right] \\ &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0^T B_* x - V_1^{*T} B_* x) x^T B_*^T \right]. \end{aligned}$$

1104 Therefore, let $V_0 = [V_{com}^{*T} \quad \mathbf{0}^T]^T$. It follows that:

$$\begin{aligned} \mathbb{E} \left[\left. \frac{\partial L}{\partial V} \right|_{\substack{V=V_0 \\ B=B_*}} \right] &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0^T B_* x - V_1^{*T} B_* x) x^T B_*^T \right] \\ &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[((V_0 - V_1^*)^T B_* x) x^T B_*^T \right] \\ &= (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{x \sim \mathcal{D}_1} [xx^T] \right) B_*^T \\ &= (V_0 - V_1^*)^T B_* B_*^T && \text{(second moment is identity)} \\ &= (V_0 - V_1^*)^T. && (B_* \text{ has orthonormal rows}) \end{aligned}$$

1105 Let $B_* = \begin{bmatrix} b_1^{*T} \\ \vdots \\ b_m^{*T} \\ b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} \end{bmatrix}$. Then, similarly, it can be shown that:

$$\begin{aligned} \mathbb{E} \left[\left. \frac{\partial L}{\partial b_j} \right|_{\substack{V=V_0 \\ B=B_*}} \right] &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0^T B_* x - V_1^{*T} B_* x) (V_0)_j x^T \right] \\ &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0)_j ((V_0 - V_1^*)^T B_* x) x^T \right] \\ &= (V_0)_j (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{x \sim \mathcal{D}_1} [xx^T] \right) \\ &= (V_0)_j (V_0 - V_1^*)^T B_*. && \text{(second moment is identity)} \end{aligned}$$

1106 Here, $(V_0)_j$ is the j -th element of the vector V_0 . For learning rate η , one step of gradient descent is:

$$V_{FT} = V_0 - \eta \left(\frac{\partial L}{\partial V} \Big|_{V=V_0, B=B_*} \right)^T$$

$$b_j^{FT} = b_j^* - \eta \left(\frac{\partial L}{\partial b_j} \Big|_{V=V_0, B_0=B_*} \right)^T.$$

1107 These two equations can be further refined as:

$$\mathbb{E}[V_{FT}] = [V_{com}^{*T} \quad \mathbf{0}^T]^T - \eta(V_0 - V_1^*) = [V_{com}^{*T} \quad \eta\lambda e_1^T]^T$$

$$\mathbb{E}[b_j^{FT}] = b_j^* - \eta \left(\frac{\partial L}{\partial b_j} \Big|_{V=V_0, B_0=B_*} \right)^T = b_j^* - \eta \left((V_0)_j (V_0 - V_1^*)^T B_* \right)^T$$

$$= b_j^* - \eta\lambda \left((V_0)_j [\mathbf{0}^T \quad -e_1^T] B_* \right)^T = b_j^* + \eta\lambda (V_0)_j b_{m+1}^*.$$

1108 Therefore, we have:

$$\mathbb{E}[B_{FT}] = \begin{bmatrix} b_1^{*T} + \eta\lambda(V_0)_1 b_{m+1}^{*T} \\ \vdots \\ b_m^{*T} + \eta\lambda(V_0)_m b_{m+1}^{*T} \\ b_{m+1}^{*T} + \eta\lambda(V_0)_{m+1} b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} + \eta\lambda(V_0)_{m+C} b_{m+1}^{*T} \end{bmatrix}$$

$$= \begin{bmatrix} b_1^{*T} + \eta\lambda(V_0)_1 b_{m+1}^{*T} \\ \vdots \\ b_m^{*T} + \eta\lambda(V_0)_m b_{m+1}^{*T} \\ b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} \end{bmatrix} = \begin{bmatrix} b_1^{*T} + \eta\lambda(V_{com}^*)_1 b_{m+1}^{*T} \\ \vdots \\ b_m^{*T} + \eta\lambda(V_{com}^*)_m b_{m+1}^{*T} \\ b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} \end{bmatrix}.$$

1109 Similarly, if the fine-tuning is done over the data of client i , we would have:

$$\mathbb{E}[b_j^{FT}] = b_j^* + \eta\lambda(V_0)_j b_{m+i}^*,$$

1110 which concludes the proof. \square

1111 *Proof of Theorem 4.4.* We assume that the pre-trained model perfectly captures the feature extractor
 1112 matrix B_* , and its linear head represents the common part shared across all clients, excluding any
 1113 client-specific components of the ground-truth function. Thus, $B_0 = B_*$ and $V_0 = [V_{com}^{*T} \quad \mathbf{0}^T]^T$.
 1114 In this setting, we analyze the effects of LP-FT and FT on the model parameters. For both LP-FT and
 1115 FT, we determine the parameters after fine-tuning, compute the global loss, and then compare these
 1116 global losses.

1117 W.L.O.G. we assume that we are doing the fine-tuning over the local data of client 1. First, we study
 1118 FT.

1119 It can be shown that:

$$\mathcal{L}_G(V_{FT}, B_{FT}) = \frac{1}{C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[\frac{1}{2} (V_{FT}^T B_{FT} x - V_i^{*T} B_* x)^2 \right]$$

$$\begin{aligned}
&= \frac{1}{2C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[(B_{FT}^T V_{FT} - B_*^T V_i^*)^T x x^T (B_{FT}^T V_{FT} - B_*^T V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left[\mathbb{E}_{x \sim \mathcal{D}_i} x x^T \right] (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T (B_{FT}^T V_{FT} - B_*^T V_i^*) \quad (\text{second moment is } I_d) \\
&= \frac{1}{2C} \sum_{i \in [C]} \|(B_{FT}^T V_{FT} - B_*^T V_i^*)\|_2^2. \tag{1}
\end{aligned}$$

1120 We have:

$$\begin{aligned}
B_*^T V_i^* &= \sum_{j=1}^m (V_{com}^*)_j b_j^* + \lambda b_{m+i}^* \\
B_{FT}^T V_{FT} &= \sum_{j=1}^m (V_{com}^*)_j b_j^* + \sum_{j=1}^m \eta \lambda (V_{com}^*)_j^2 b_{m+1}^* + \eta \lambda b_{m+1}^*.
\end{aligned}$$

1121 Therefore, we can obtain:

$$(B_{FT}^T V_{FT} - B_*^T V_i^*) = \lambda \left(\sum_{j=1}^m \eta (V_{com}^*)_j^2 b_{m+1}^* + \eta b_{m+1}^* - b_{m+i}^* \right).$$

1122 For $i \neq 1$, we have:

$$\begin{aligned}
&(B_{FT}^T V_{FT} - B_*^T V_i^*)^T (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \lambda^2 \left(\sum_{j=1}^m \eta (V_{com}^*)_j^2 b_{m+1}^* + \eta b_{m+1}^* - b_{m+i}^* \right)^T \left(\sum_{j=1}^m \eta (V_{com}^*)_j^2 b_{m+1}^* + \eta b_{m+1}^* - b_{m+i}^* \right) \\
&= \lambda^2 \left(\left(\eta + \eta \sum_{j=1}^m (V_{com}^*)_j^2 \right)^2 + 1 \right). \quad (\text{rows of } B_* \text{ are orthonormal})
\end{aligned}$$

1123 For $i = 1$, we have:

$$\begin{aligned}
&(B_{FT}^T V_{FT} - B_*^T V_i^*)^T (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \lambda^2 \left(\sum_{j=1}^m \eta (V_{com}^*)_j^2 b_{m+1}^* + \eta b_{m+1}^* - b_{m+i}^* \right)^T \left(\sum_{j=1}^m \eta (V_{com}^*)_j^2 b_{m+1}^* + \eta b_{m+1}^* - b_{m+i}^* \right) \\
&= \lambda^2 \left(\eta + \eta \sum_{j=1}^m (V_{com}^*)_j^2 - 1 \right)^2. \quad (\text{rows of } B_* \text{ are orthonormal})
\end{aligned}$$

1124 Combining these with (1), we can conclude:

$$\begin{aligned}
\mathcal{L}_G(V_{FT}, B_{FT}) &= \frac{1}{2C} \sum_{i \in [C]} \|(B_{FT}^T V_{FT} - B_*^T V_i^*)\|_2^2 \\
&= \frac{\lambda^2}{2C} \left(\left(\eta + \eta \sum_{j=1}^m (V_{com}^*)_j^2 - 1 \right)^2 + (C-1) \left(\left(\eta + \eta \sum_{j=1}^m (V_{com}^*)_j^2 \right)^2 + 1 \right) \right). \tag{2}
\end{aligned}$$

1125 We can follow the same procedure for LP-FT as follows:

1126 LP step starts from $B_0 = B_*$ and $V_0 = [V_{com}^*{}^T \quad \mathbf{0}]^T$. From proof of Lemma 4.3, we know that

1127 $\mathbb{E} \left[\frac{\partial L}{\partial V} \Big|_{\substack{V=V_0 \\ B=B_*}} \right] = (V_0 - V_1^*)^T$. Therefore, after one iteration of LP, we have:

$$V_{LP} = V_0 - \eta \left(\frac{\partial L}{\partial V} \Big|_{\substack{V=V_0 \\ B=B_*}} \right)^T$$

1128 After I iterations we have:

$$V_{LP} = [V_{com}^*{}^\top \quad \lambda \alpha e_1^\top]^\top \quad \text{for} \quad \alpha = (1 - (1 - \eta)^I)$$

1129 Let the initial parameters after one step of LP and before the FT step be

$$B_{LP} = B_*, \quad V_{LP} = [V_{com}^*{}^\top \quad \lambda \alpha e_1^\top]^\top, \quad \alpha \in [0, 1].$$

1130 Assume the first client ($i = 1$) runs a single full-batch gradient-descent step with learning-rate η on
 1131 its local MSE loss. Let V_{LPFT} and B_{LPFT} be the resulting linear head and feature-extractor matrix
 1132 and $(b_j^{LPFT})^\top$ its j -th row after the final FT step. Then

$$\begin{aligned} \mathbb{E}[V_{LPFT}] &= [V_{com}^*{}^T \quad \lambda(\alpha + \eta(1 - \alpha))e_1^T]^\top \\ \mathbb{E}[b_j^{LPFT}] &= b_j^* - \eta \left(\frac{\partial L}{\partial b_j} \Big|_{\substack{V=V_{LP} \\ B_{LP}=B_*}} \right)^T = b_j^* - \eta \left((V_{LP})_j (V_{LP} - V_1^*)^T B_* \right)^T \\ &= b_j^* - \eta \lambda (1 - \alpha) \left((V_{LP})_j [\mathbf{0}^T \quad -e_1^T] B_* \right)^T = b_j^* + \eta \lambda (1 - \alpha) (V_{LP})_j b_{m+1}^*. \end{aligned}$$

1133 This steps can be directly obtained exactly as we derived $\mathbb{E}[V_{FT}]$ and $\mathbb{E}[b_j^{FT}]$ for FT case.

1134 Similar to computing $\mathcal{L}_G(V_{FT}, B_{FT})$, we find $\mathcal{L}_G(V_{LPFT}, B_{LPFT})$ as follows:

1135 Writing $S := \sum_{j=1}^m (V_{com}^*)_j^2 = \|V_{com}^*\|_2^2$ and

$$A_\alpha := \eta(1 - \alpha)S + [\alpha + \eta(1 - \alpha)][1 + \eta\lambda^2\alpha(1 - \alpha)],$$

1136 the expected global loss

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) := \frac{1}{C} \sum_{i=1}^C \mathbb{E}_{x \sim \mathcal{D}_i} \left[\frac{1}{2} (V_{LPFT}^\top B_{LPFT} x - V_i^*{}^\top B_* x)^2 \right]$$

1137 satisfies

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) = \frac{\lambda^2}{2C} \left[(A_\alpha - 1)^2 + (C - 1)(A_\alpha^2 + 1) \right].$$

1138 this is because the LP-FT loss can be obtained similar to FT loss as in (1),

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) = \frac{1}{2C} \sum_{i=1}^C \|B_{LPFT}^\top V_{LPFT} - B_*^\top V_i^*\|_2^2.$$

1139 First, it can be shown that:

$$B_{LPFT}^\top V_{LPFT} = \sum_{j=1}^m (V_{com}^*)_j b_j^* + \lambda A_\alpha b_{m+1}^*.$$

1140 For each client $i \in [C]$, $B_*^\top V_i^* = \sum_{j=1}^m (V_{com}^*)_j b_j^* + \lambda b_{m+i}^*$. Therefore

$$\Delta_i := B_{LPFT}^\top V_{LPFT} - B_*^\top V_i^* = \lambda [A_\alpha b_{m+1}^* - b_{m+i}^*].$$

1141 Because the rows of B_* are orthonormal,

$$\|\Delta_i\|_2^2 = \begin{cases} \lambda^2 (A_\alpha - 1)^2, & i = 1, \\ \lambda^2 (A_\alpha^2 + 1), & i \neq 1. \end{cases}$$

1142 Substituting the above norms in the global loss,

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) = \frac{\lambda^2}{2C} \left[(A_\alpha - 1)^2 + (C - 1)(A_\alpha^2 + 1) \right].$$

1143 which is the stated result.

1144 Now, we compare $\mathcal{L}_G(V_{LPFT}, B_{LPFT})$ and $\mathcal{L}_G(V_{FT}, B_{FT})$ by computing the difference:

$$\begin{aligned} \Delta &:= \mathcal{L}_G(V_{LPFT}, B_{LPFT}) - \mathcal{L}_G(V_{FT}, B_{FT}) \\ &= \frac{\lambda^2}{2C} \left[(A_\alpha - 1)^2 - (\eta(1 + S) - 1)^2 + (C - 1)(A_\alpha^2 - \eta^2(1 + S)^2) \right] \\ &= \frac{\lambda^2}{2C} \left[C(A_\alpha^2 - \eta^2(1 + S)^2) + 2(\eta(1 + S) - A_\alpha) \right] \\ &= \frac{\lambda^2}{2C} (A_\alpha - \eta(1 + S)) (C(A_\alpha + \eta(1 + S)) - 2) \\ &\leq 0. \end{aligned}$$

1145 This is because, at convergence of the linear probing (LP) step, $\alpha \rightarrow 1$, and the product

$$(A_\alpha - \eta(1 + S)) (C(A_\alpha + \eta(1 + S)) - 2)$$

1146 becomes non-positive. Therefore, we have:

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) \leq \mathcal{L}_G(V_{FT}, B_{FT}).$$

1147 □

1148 *Proof of Theorem 4.5.* W.L.O.G. we assume that the local fine-tuning is performed on the data of the
1149 first client. Initially, one step of linear probing is conducted with the fixed feature extractor B_* . After
1150 this step, the linear head V_{LP} will converge to V_1^* . This is because we know that:

$$\arg \min_v \|\mathbf{X}_1 B_0^\top v - \mathbf{X}_1 B_*^\top v_*\|_2^2 = \left(B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_0^\top \right)^{-1} B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_*^\top v_*,$$

1151 where \mathbf{X}_1 is the $n \times d$ matrix including data of n individuals. Since the fine-tuning is on the data of
1152 the client 1 (local data), we have:

$$V_{LP} = \left(B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_0^\top \right)^{-1} B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_*^\top V_1^*.$$

1153 Therefore, we have:

$$\begin{aligned} V_{LP} &= \left(B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_0^\top \right)^{-1} B_0 \mathbf{X}_1^\top \mathbf{X}_1 B_*^\top V_1^* \\ &= \left(B_* \mathbf{X}_1^\top \mathbf{X}_1 B_*^\top \right)^{-1} B_* \mathbf{X}_1^\top \mathbf{X}_1 B_*^\top V_1^* \\ &= V_1^*. \end{aligned}$$

1154 This part is identical to the initial part of the proof of Theorem 4.4. Since at the beginning of the
1155 fine-tuning (FT) step, we have the perfect B_* and V_1^* for the local client 1, and FT is performed on
1156 the data of the same client, we can conclude that after one step of FT following LP, the parameters
1157 will remain unchanged. Specifically, we have $V_{LPFT} = V_1^* = [V_{com}^{*T} \quad \lambda e_1^T]^\top$ and $B_{LPFT} = B_*$.

1158 For the performance on the global data, we have:

$$\begin{aligned} \mathcal{L}_G(V_{LPFT}, B_{LPFT}) &= \frac{1}{C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[\frac{1}{2} (V_{LPFT}^\top B_{LPFT} x - V_i^{*T} B_* x)^2 \right] \\ &= \frac{1}{C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[\frac{1}{2} (V_1^{*T} B_* x - V_i^{*T} B_* x)^2 \right] \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[(B_*^T V_1^* - B_*^T V_i^*)^T x x^T (B_*^T V_1^* - B_*^T V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(B_*^T V_1^* - B_*^T V_i^*)^T \mathbb{E}_{x \sim \mathcal{D}_i} [x x^T] (B_*^T V_1^* - B_*^T V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(\mathbb{E}_{x \sim \mathcal{D}_i} [x x^T] \right) B_*^T (V_1^* - V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(\mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [(e_i + \epsilon z)(e_i + \epsilon z)^T] \right) B_*^T (V_1^* - V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(e_i e_i^T + \epsilon^2 \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [z z^T] \right) B_*^T (V_1^* - V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(e_i e_i^T + \epsilon^2 I_d \right) B_*^T (V_1^* - V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(e_i e_i^T \right) B_*^T (V_1^* - V_i^*) \right] \\
&\quad + \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T B_* \left(\epsilon^2 I_d \right) B_*^T (V_1^* - V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \left[(V_1^* - V_i^*)^T (B_*)_{:,i} (B_*)_{:,i}^T (V_1^* - V_i^*) \right] \quad ((B_*)_{:,i} \text{ } i\text{-th column of } B_*) \\
&\quad + \frac{1}{2C} \sum_{i \in [C]} \left[\epsilon^2 (V_1^* - V_i^*)^T (V_1^* - V_i^*) \right] \quad (B_* \text{ has orthonormal rows}) \\
&= \frac{\lambda^2}{2C} \sum_{i \in [C]} \left[\left((B_*)_{m+1,i} - (B_*)_{m+i,i} \right)^2 \right] + \frac{1}{2C} \epsilon^2 \sum_{i \in [C]} \left[\|(V_1^* - V_i^*)\|_2^2 \right] \\
&= \frac{\lambda^2}{2C} \sum_{i \in [C]} \left[\left((B_*)_{m+1,i} - (B_*)_{m+i,i} \right)^2 \right] + \frac{\lambda^2 (C-1)}{C} \epsilon^2. \tag{3}
\end{aligned}$$

1159 We want to analyze the fine-tuning (FT) method, focusing on the effect of initial parameters. We
1160 perform one pass through the entire dataset to simulate the complete fine-tuning process. Consider
1161 the Mean Squared Error (MSE) loss function with parameters V and B , where B is represented as
1162 follows:

$$B = \begin{bmatrix} b_1^T \\ \vdots \\ b_m^T \\ b_{m+1}^T \\ \vdots \\ b_{m+C}^T \end{bmatrix},$$

1163 where $b_i^T \in \mathbb{R}^{1 \times d}$ denotes the i -th row of matrix B , and $m + C = k$.

1164 To apply one step of gradient descent, we need to compute the gradient of the loss function with
1165 respect to V , b_1 , b_2 , \dots , b_{m+C} , and then perform one update step.

1166 Let $V_0 = [V_{com}^* \quad \mathbf{0}^T]^T$. It follows that:

$$\mathbb{E} \left[\frac{\partial L}{\partial V} \Big|_{V=V_0, B=B_*} \right] = \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0^T B_* x - V_1^{*T} B_* x) x^T B_*^T \right]$$

$$\begin{aligned}
&= \mathbb{E}_{x \sim \mathcal{D}_1} \left[((V_0 - V_1^*)^T B_* x) x^T B_*^T \right] \\
&= (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{x \sim \mathcal{D}_1} [x x^T] \right) B_*^T \\
&= (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [(e_1 + \epsilon z)(e_1 + \epsilon z)^T] \right) B_*^T \\
&= (V_0 - V_1^*)^T B_* (e_1 e_1^T + \epsilon^2 I_d) B_*^T \\
&= (V_0 - V_1^*)^T B_* (e_1 e_1^T) B_*^T + (V_0 - V_1^*)^T B_* (\epsilon^2 I_d) B_*^T \\
&= (V_0 - V_1^*)^T B_* (e_1 e_1^T) B_*^T + \epsilon^2 (V_0 - V_1^*)^T \quad (B_* \text{ has orthonormal rows}) \\
&= (V_0 - V_1^*)^T \left((B_*)_{:,1} (B_*)_{:,1}^T \right) + \epsilon^2 (V_0 - V_1^*)^T \quad ((B_*)_{:,1} \text{ is first column of } B_*) \\
&= \left(-\lambda(B_*)_{m+1,1} \right) (B_*)_{:,1}^T + \epsilon^2 (V_0 - V_1^*)^T.
\end{aligned}$$

1167 Let $B_* = \begin{bmatrix} b_1^{*T} \\ \vdots \\ b_m^{*T} \\ b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} \end{bmatrix}$. Then, it can be shown that:

$$\begin{aligned}
\mathbb{E} \left[\frac{\partial L}{\partial b_j} \Big|_{\substack{V=V_0 \\ B=B_*}} \right] &= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0^T B_* x - V_1^{*T} B_* x) (V_0)_j x^T \right] \\
&= \mathbb{E}_{x \sim \mathcal{D}_1} \left[(V_0)_j \left((V_0 - V_1^*)^T B_* x \right) x^T \right] \\
&= (V_0)_j (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{x \sim \mathcal{D}_1} [x x^T] \right) \\
&= (V_0)_j (V_0 - V_1^*)^T B_* \left(\mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [(e_1 + \epsilon z)(e_1 + \epsilon z)^T] \right) \\
&= (V_0)_j (V_0 - V_1^*)^T B_* (e_1 e_1^T + \epsilon^2 I_d) \\
&= (V_0)_j (V_0 - V_1^*)^T B_* (e_1 e_1^T) + (V_0)_j (V_0 - V_1^*)^T B_* (\epsilon^2 I_d) \\
&= (V_0)_j (V_0 - V_1^*)^T B_* (e_1 e_1^T) + \epsilon^2 (V_0)_j (V_0 - V_1^*)^T B_*.
\end{aligned}$$

1168 Here, $(V_0)_j$ is the j -th element of the vector V_0 . For learning rate η , one step of gradient descent can
1169 be:

$$\begin{aligned}
V_{FT} &= V_0 - \eta \left(\frac{\partial L}{\partial V} \Big|_{\substack{V=V_0 \\ B=B_*}} \right)^T \\
b_j^{FT} &= b_j^* - \eta \left(\frac{\partial L}{\partial b_j} \Big|_{\substack{V=V_0 \\ B=B_*}} \right)^T.
\end{aligned}$$

1170 These two equations can be further refined as:

$$\mathbb{E}[V_{FT}] = [V_{com}^{*T} \quad \mathbf{0}^T]^T - \eta \left(-\lambda(B_*)_{m+1,1} (B_*)_{:,1} + \epsilon^2 (V_0 - V_1^*) \right)$$

$$\begin{aligned}
&= \begin{bmatrix} V_{com}^* \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \eta\lambda\epsilon^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \eta\lambda(B_*)_{m+1,1}(B_*)_{:,1} \\
\mathbb{E}[b_j^{FT}] &= b_j^* - \eta \left((V_0)_j (V_0 - V_1^*)^T B_* \left(e_1 e_1^T \right) + \epsilon^2 (V_0)_j (V_0 - V_1^*)^T B_* \right)^T \\
&= b_j^* + \begin{bmatrix} \eta\lambda(V_0)_j (B_*)_{m+1,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \eta\lambda(V_0)_j \epsilon^2 b_{m+1}^*.
\end{aligned}$$

1171 Therefore, we have:

$$\begin{aligned}
\mathbb{E}[B_{FT}] &= \begin{bmatrix} b_1^{*T} + \eta\lambda(V_0)_1 \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_1 (B_*)_{m+1,1} e_1^T \\ \vdots \\ b_m^{*T} + \eta\lambda(V_0)_m \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_m (B_*)_{m+1,1} e_1^T \\ b_{m+1}^{*T} + \eta\lambda(V_0)_{m+1} \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_{m+1} (B_*)_{m+1,1} e_1^T \\ \vdots \\ b_{m+C}^{*T} + \eta\lambda(V_0)_{m+C} \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_{m+C} (B_*)_{m+1,1} e_1^T \end{bmatrix} \\
&= \begin{bmatrix} b_1^{*T} + \eta\lambda(V_0)_1 \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_1 (B_*)_{m+1,1} e_1^T \\ \vdots \\ b_m^{*T} + \eta\lambda(V_0)_m \epsilon^2 b_{m+1}^{*T} + \eta\lambda(V_0)_m (B_*)_{m+1,1} e_1^T \\ b_{m+1}^{*T} \\ \vdots \\ b_{m+C}^{*T} \end{bmatrix}.
\end{aligned}$$

1172 It can be shown that:

$$\begin{aligned}
\mathcal{L}_G(V_{FT}, B_{FT}) &= \frac{1}{C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[\frac{1}{2} (V_{FT}^T B_{FT} x - V_i^{*T} B_* x)^2 \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} \mathbb{E}_{x \sim \mathcal{D}_i} \left[(B_{FT}^T V_{FT} - B_*^T V_i^*)^T x x^T (B_{FT}^T V_{FT} - B_*^T V_i^*) \right] \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left[\mathbb{E}_{x \sim \mathcal{D}_i} x x^T \right] (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left(e_i e_i^T + \epsilon^2 I_d \right) (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left(e_i e_i^T \right) (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&\quad + \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left(\epsilon^2 I_d \right) (B_{FT}^T V_{FT} - B_*^T V_i^*) \\
&= \frac{1}{2C} \sum_{i \in [C]} (B_{FT}^T V_{FT} - B_*^T V_i^*)^T \left(e_i e_i^T \right) (B_{FT}^T V_{FT} - B_*^T V_i^*)
\end{aligned}$$

$$+ \epsilon^2 \frac{1}{2C} \sum_{i \in [C]} \|(B_{FT}^T V_{FT} - B_*^T V_i^*)\|_2^2. \quad (4)$$

1173 We have:

$$\begin{aligned} B_*^T V_i^* &= \sum_{j=1}^m (V_{com}^*)_j b_j^* + \lambda b_{m+i}^* \\ B_{FT}^T V_{FT} &= \eta \lambda \epsilon^2 \sigma^2 b_{m+1}^* + \sum_{j=m+1}^{m+C} \left(\eta \lambda (B_*)_{m+1,1} (B_*)_{j,1} \right) b_j^* \\ &+ \sum_{j=1}^m \left((V_{com}^*)_j + \eta \lambda (B_*)_{m+1,1} (B_*)_{j,1} \right) \left(b_j^* + \eta \lambda (V_{com}^*)_j \epsilon^2 \sigma^2 b_{m+1}^* + \eta \lambda (V_{com}^*)_j (B_*)_{m+1,1} e_1 \right). \end{aligned}$$

1174 Therefore, we can obtain:

$$\begin{aligned} B_{FT}^T V_{FT} - B_*^T V_i^* &= \sum_{j=1}^m (V_{com}^*)_j \left(b_j^* + \eta \lambda (V_{com}^*)_j \epsilon^2 \sigma^2 b_{m+1}^* + \eta \lambda (V_{com}^*)_j (B_*)_{m+1,1} e_1 \right) \\ &+ \eta \lambda \epsilon^2 \sigma^2 b_{m+1}^* - \lambda b_{m+i}^* + \sum_{j=m+1}^{m+C} \left(\eta \lambda (B_*)_{m+1,1} (B_*)_{j,1} \right) b_j^* \\ &+ \sum_{j=1}^m \left(\eta \lambda (B_*)_{m+1,1} (B_*)_{j,1} \right) \left(b_j^* + \eta \lambda (V_{com}^*)_j \epsilon^2 \sigma^2 b_{m+1}^* + \eta \lambda (V_{com}^*)_j (B_*)_{m+1,1} e_1 \right). \end{aligned} \quad (5)$$

$$(6)$$

1175 From equation (3), we observe that $\mathcal{L}_G(V_{LPFT}, B_{LPFT})$ is a monotonically decreasing function of
 1176 λ and as λ approaches zero, $\mathcal{L}_G(V_{LPFT}, B_{LPFT})$ also converges to zero. In contrast, combining
 1177 equations (4) and (5), we find that $\mathcal{L}_G(V_{FT}, B_{FT})$ does not converge to zero as λ approaches zero
 1178 due to the presence of constant terms independent of λ . Therefore, it follows that there always exists
 1179 a threshold λ^* such that for all $\lambda \leq \lambda^*$:

$$\mathcal{L}_G(V_{LPFT}, B_{LPFT}) \leq \mathcal{L}_G(V_{FT}, B_{FT}).$$

1180

□

1181 F Empirical Performance of LP-FT and FT Under Theorem 4.5 Conditions

1182 To give a better understanding of Theorem 4.5, we give a simple visualization of two randomly
 1183 generated data-generating functions for different clients and compute the global loss of LP-FT and
 1184 FT based on equations (3) and (4).

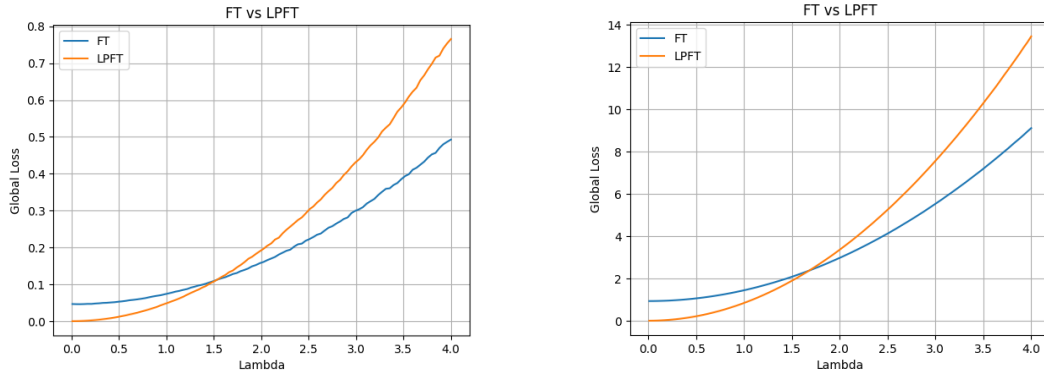


Figure 6: (a) Global loss of LP-FT and FT as a function of the heterogeneity parameter λ , with $\eta = 0.1$, $\epsilon = 0.1$, matrix B_* as a 10×20 random matrix, and number of clients $C = 5$. (b) Global loss of LP-FT and FT as a function of the heterogeneity parameter λ , with $\eta = 0.1$, $\epsilon = 1$, matrix B_* as a 10×20 random matrix, and number of clients $C = 5$.

These examples illustrate, within the theoretical setting of Sec. 4.2, the behavior of the loss functions for LP-FT and FT with a randomly generated labeling function $y = V_i^{*T} B_* x$, a fixed learning rate η , noise parameter ϵ , and a fixed number of clients C . To compute this, we generated 1000 random matrices B_* and 1000 randomly chosen linear heads V_i^* as ground-truth labeling functions, ensuring they adhere to the theoretical assumptions. Using equations (3) and (4), we calculated the average loss of LP-FT and FT across these random trials.

As shown in Fig. 6, there exists a threshold λ^* such that when $\lambda \leq \lambda^*$, LP-FT consistently outperforms FT. While this is a simplified example with a fixed number of clients, learning rate, noise parameter, and dimensionality of the ground-truth parameters B_* and V_i^* , the observed trend remains similar across different parameter settings. The purpose of this figure is to provide an intuitive understanding of Theorem 4.5 in a controlled, simplified context. More comprehensive experiments in Sec. D.3 demonstrate that LP-FT globally outperforms FT across a broader range of heterogeneity levels in real-world settings.

G Computational Overhead of LP-FT Relative to FT

In this section, we want to study the computation cost of adding one step of linear probing (LP) to the full-fine tuning (FT) to see how this additional LP step affects computational cost. Suppose the dimension of the output of the feature extractor layer (input of the linear head) is d and the dimension of the output of the linear head is m . In fact, the linear head will be a $d \times m$ linear layer. We assume having n samples. We want to see what is the computational cost of fine-tuning this linear head.

To estimate the computational cost of training a linear neural network layer with d inputs, m outputs, and n samples, we analyze the steps involved:

1. **Forward Pass:** A linear neural network computes outputs as:

$$Y = XW,$$

where:

- $X \in \mathbb{R}^{n \times d}$ is the input matrix (with n samples, each of dimension d),
- $W \in \mathbb{R}^{d \times m}$ is the weight matrix,
- $Y \in \mathbb{R}^{n \times m}$ is the output matrix.

The cost of this matrix multiplication is $O(ndm)$.

2. **Backward Pass (Gradient Computation):** To update W , the gradient of the loss \mathcal{L} with respect to W is computed. We know that:

$$\nabla_W \mathcal{L} = X^T (\nabla_Y \mathcal{L})$$

Therefore, computing $(\nabla_W \mathcal{L})$ involves:

- Computing the gradient of the loss with respect to the outputs Y , which has a cost of $O(nk)$,
- matrix multiplication $\nabla_W \mathcal{L} = X^T (\nabla_Y \mathcal{L})$ which involves a matrix multiplication with a cost of $O(ndm)$.

3. **Weight Update:** If using gradient descent, the cost of updating the weights is $O(dm)$.

Total Computational Cost: The total cost for one forward and backward pass through the data is dominated by $O(ndm)$, which accounts for both forward propagation and gradient computation. If the training involves multiple epochs, the total cost scales as:

$$O(e \cdot ndm),$$

where e is the number of epochs.

Broader Impact

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

1228 **Limitation**

1229 While our theoretical analysis is limited to concept and combined concept-covariate shifts under
1230 simplified assumptions (e.g., linear models and shared feature extractors), this does not significantly
1231 undermine our contributions. Extensive empirical results across seven diverse datasets demonstrate
1232 that LP-FT consistently outperforms baseline methods in both personalization and generalization,
1233 even in complex, real-world scenarios. This suggests that the core insights behind LP-FT—such
1234 as mitigating feature distortion through phased fine-tuning—generalize well beyond the theoretical
1235 scope and offer practical value for robust federated personalization.