FedDr+: Stabilizing Dot-regression with Global Feature Distillation for Federated Learning

Seongyoon Kim
Dept. ISysE, KAIST
Daejeon, Republic of Korea
curisam@kaist.ac.kr

Minchan Jeong
KAIST AI
Seoul, Republic of Korea
mcjeong@kaist.ac.kr

Sungnyun Kim
KAIST AI
Seoul, Republic of Korea
ksn4397@kaist.ac.kr

Sungwoo Cho
KAIST AI
Seoul, Republic of Korea
peter8526@kaist.ac.kr

Sumyeong Ahn[†*]
CSE, Michigan State University
East Lansing, United States
sumyeong@msu.edu

Se-Young Yun[†]
KAIST AI
Seoul, Republic of Korea
yunseyoung@kaist.ac.kr

ABSTRACT

Federated Learning (FL) has emerged as a pivotal framework for developing effective global models across clients with heterogeneous, non-iid data distribution. A key challenge in FL is client drift, where data heterogeneity impedes the aggregation of scattered knowledge. Recent studies have tackled client drift by identifying significant divergence in the last classifier layer. To mitigate this divergence, strategies such as freezing classifier weights and aligning the feature extractor accordingly have proven effective. However, while local alignment between classifier and feature extractor is crucial in FL, it may cause the model to overemphasize observed classes within each client. Our objective is twofold: (1) enhancing local alignment while (2) preserving the representation of unseen class samples. We introduce a novel algorithm named **FedDr+**, which enhances local model alignment using dot-regression loss. **FedDr+** freezes the classifier as a simplex ETF to align features and improves aggregated global models through a feature distillation mechanism to retain information about unseen/missing classes. Empirical evidence demonstrates that our algorithm surpasses existing methods that use a frozen classifier to enhance alignment across diverse distributions.

KEYWORDS

Federated learning, Dot-regression, Knowledge distillation

ACM Reference Format:
Seongyoon Kim, Minchan Jeong, Sungnyun Kim, Sungwoo Cho, Sumyeong Ahn, and Se-Young Yun. 2024. FedDr+: Stabilizing Dot-regression with Global Feature Distillation for Federated Learning. In *FedKDD '24, August 26, 2024, Barcelona, Spain.*. ACM, New York, NY, USA, 13 pages.

1 INTRODUCTION

Federated Learning (FL) [13, 36, 38] is a privacy-aware distributed learning strategy that employs data from multiple clients while ensuring their data privacy. A foundational method in FL, known as FedAvg [36], involves four iterative phases: (1) distributing a global model to clients, (2) training local models using each client's private dataset, (3) transmitting the locally trained models back to the server, and (4) aggregating these models. This method effectively protects privacy without requiring the transmission of raw data to the server. However, a significant challenge in FL is data heterogeneity, called *non-iidness*, which refers to the different underlying data distribution across clients. Such variance can cause *client drift* during training, obstructing the convergence of the aggregated model and significantly reducing its effectiveness.

To address client drift in non-iid scenarios, recent works [8, 10, 33, 38] have identified that the last classifier layer in neural networks is particularly vulnerable to this issue. Hence, they suggest strategies that freeze the classifier while updating only the feature extractor. These approaches aim to enhance the *alignment* between the frozen classifier and the output from the feature extractor. For instance, FedBABU [38] employs various classifier initialization techniques, keeping it fixed during the training of the feature extractor. The methods proposed in [8, 10, 18, 33, 44] utilize more robust initialization, the Equiangular Tight Frame (ETF) classifier [39], to replace traditional random initialization and improve the local alignment strategy.

A frozen classifier is also extensively explored in other research areas, such as class imbalance [45] and class incremental learning [46], with a consistent objective similar to aforementioned FL studies—enhancing alignment. Recently, these fields have advanced by introducing and utilizing a novel type of loss, called dot-regression loss $\mathcal{L}_{\text{DR}}$, which aims to achieve alignment rapidly. In summary, $\mathcal{L}_{\text{DR}}$ originates from the decomposition analysis of cross-entropy (CE) loss, which includes *pulling* and *pushing*.

As suggested in [45], the *pulling* component is a force that attracts features to the target class, whereas the *pushing* component is a force that drives features away from other non-target classes. $\mathcal{L}_{\text{DR}}$ discards the *pushing* component, as it slows down convergence to the desired alignment (refer to Figure 1).

Following the advancement of leveraging the frozen classifier with dot-regression loss, we investigate the application of this loss

[*]This work was done while Sumyeong Ahn was at KAIST.

FedKDD '24, August 26, 2024, Barcelona, Spain.

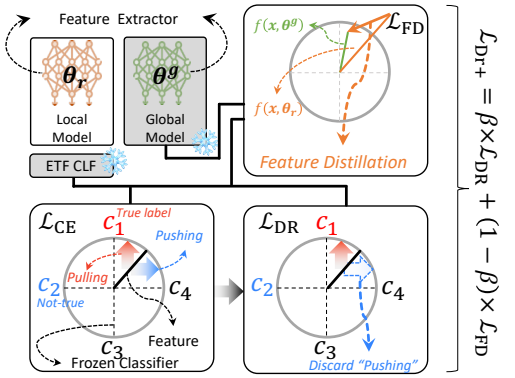


Figure 1: Overview of the proposed method, FedDr+ trained with $\mathcal{L}_{\text{Dr+}}$. To enhance the local alignment, we employ dot-regression loss $\mathcal{L}_{\text{DR}}$, which discards the pushing term of cross-entropy loss, and propose a feature distillation $\mathcal{L}_{\text{FD}}$ to preserve the knowledge imbued in the global model.

to FL. However, our findings indicate that dot-regression loss does not necessarily lead to sufficient performance improvement of the aggregated server-side model, although it enhances *local alignment* as intended. We observe that this drawback stems from the handling of unseen class samples. Specifically, while alignment improves for the classes in the local training dataset, it significantly deteriorates for unseen classes. This observation highlights the need to preserve the representation of unobserved classes during local training. To address this issue, we propose a training mechanism, termed **FedDr+**, that employs dot-regression loss alongside feature distillation that reduces the distance between feature vectors of local and global models.

Contributions. Our main contributions are summarized as follows:

- We find that dot-regression loss is not easily compatible with FL, although it can enhance the alignment of seen classes. The drawback comes from a significant loss of information on unseen classes, which is vital in the global model perspective. Therefore, we aim to preserve information of unseen classes within the FL system.
- To preserve global knowledge, including unseen class information while maintaining the advantages of $\mathcal{L}_{\text{DR}}$, we propose **FedDr+**, which utilizes a feature distillation when training local models. This regularizer prevents the model from focusing solely on the local alignment.
- We verify that the proposed method surpasses conventional FL algorithms under various datasets and non-iid settings.

2 PRELIMINARIES

2.1 Basic Setup of FedAvg Pipeline

Basic FL setup. Let $[N] = \{1, \dots, N\}$ denote the indices of clients, each with a unique training dataset $D^i_{\text{train}} = \{(x_m, y_m)\}_{m=1}^{|D^i_{\text{train}}|}$, where $(x_m, y_m) \sim \mathcal{D}^i$ for the i^{th} client, x_m is the input data, and $y_m \in [C]$ is the corresponding label among C classes. Importantly, FL studies predominantly address the scenario where the data distributions are heterogeneous, *i.e.,* $\mathcal{D}^i$ varies across clients. Knowledge distributed among clients is collected over R communication rounds. The general objective of FL is to train a model fit to the aggregated knowledge, $\bigcup_{i\in[N]} \mathcal{D}^i$. This objective can be seen as solving the optimization problem:

$$\min_{\boldsymbol{\Theta}=(\boldsymbol{\theta},\boldsymbol{V})} \sum_{i\in[N]} \frac{|D^i_{\text{train}}|}{\sum_{j\in[N]} |D^j_{\text{train}}|} \mathbb{E}_{(x,y)\sim\mathcal{D}^i}\Big[\mathcal{L}(x, y; \boldsymbol{\theta}, \boldsymbol{V})\Big],$$

where $\mathcal{L}$ is the instance-wise loss function, $\boldsymbol{\theta}$ is the weight parameter for the feature extractor, and $\boldsymbol{V} = [v_1, \dots, v_C] \in \mathbb{R}^{d\times C}$ is the classifier weight matrix. We use the notation $\boldsymbol{\Theta}$ to denote the entire set of model parameters.

At the beginning of each round $r \in [R]$, the server has access to only a subset of clients $\mathcal{S}_r \subset [N]$ participating in the r^{th} round. At each round r, the server transmits the global model parameters $\boldsymbol{\Theta}^g_{r-1}$ to the participating clients. Each client then updates the parameters with their private data D^i_{train} and uploads $\boldsymbol{\Theta}^i_r$ to the global server. By incorporating the locally trained weights, the server then updates the global model parameters to $\boldsymbol{\Theta}^g_r$.

FedAvg pipeline. Our study follows the FedAvg [36] framework to address the FL problem. FedAvg updates the global model parameters from locally trained parameters by aggregating these local models into $\boldsymbol{\Theta}^g_r = \sum_{i\in\mathcal{S}_r} w^i_r \boldsymbol{\Theta}^i_r$, where $w^i_r = |D^i_{\text{train}}| \,/\, \sum_{j\in\mathcal{S}_r} |D^j_{\text{train}}|$ is the importance weight of the i^{th} client.

2.2 Dot-Regression Loss for Feature Alignment

Dot-regression loss $\mathcal{L}_{\text{DR}}$. This loss [45] facilitates a faster alignment of feature vectors (penultimate layer outputs) $f(x;\boldsymbol{\theta}) \in \mathbb{R}^d$ to the true class direction of v_y, reducing the cosine angle as follows:

$$\mathcal{L}_{\text{DR}}(x, y; \boldsymbol{\theta}, \boldsymbol{V}) = \frac{1}{2}\Big(\cos\big(f(x;\boldsymbol{\theta}), v_y\big) - 1\Big)^2$$

where $\cos(\text{vec}_1, \text{vec}_2)$ denotes the cosine of the angle between two vectors $\angle(\text{vec}_1, \text{vec}_2)$.

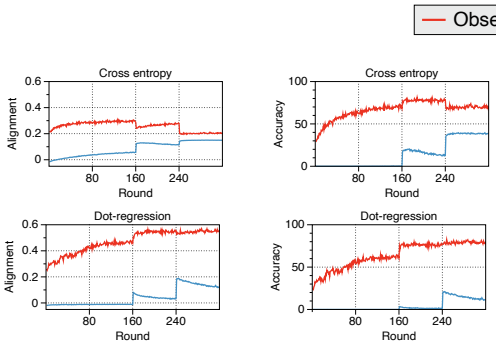
The main motivation is that the gradient of the cross-entropy (CE) loss for the feature vector can be decomposed into a *pulling* and *pushing* gradient, and recent work indicates that we can achieve better convergence by removing the pushing effect [32, 45]. The *pulling* gradient aligns $f(x;\boldsymbol{\theta})$ with v_y, while the *pushing* gradient ensures $f(x;\boldsymbol{\theta})$ does not align with v_c for all $c \neq y$ (Appendix B details the exact form of pulling and pushing gradients). Since $\mathcal{L}_{\text{DR}}$ directly attracts features to the true-class classifier, it drops the *pushing* gradient, thereby increasing the convergence speed for maximizing $\cos(f(x;\boldsymbol{\theta}), v_y)$.

Frozen ETF classifier. Since $\mathcal{L}_{\text{DR}}$ focuses on aligning feature vectors with the true-class classifier, the classifier is not required to be trained. Instead, we construct the classifier to satisfy the simplex Equiangular Tight Frame (ETF) condition, a constructive way to achieve maximum angular separation between class vectors [45, 46]. Concretely, we initialize the classifier weight $\boldsymbol{V}$ as follows and freeze it throughout training:

$$\boldsymbol{V} \longleftarrow \sqrt{\frac{C}{C-1}}\boldsymbol{U}\left(\boldsymbol{I}_C - \frac{1}{C}\mathbf{1}_C\mathbf{1}_C^\top\right),$$

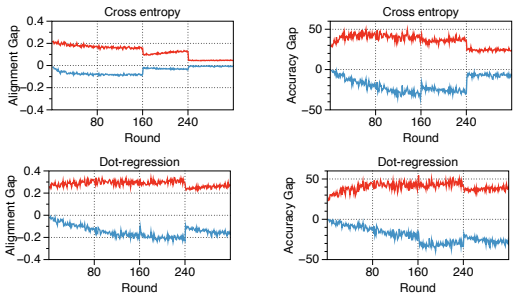
where $\boldsymbol{U} \in \mathbb{R}^{d\times C}$ is a randomly initialized orthogonal matrix. Note that each v_i in the classifier weight $\boldsymbol{V}$ satisfies $\cos(v_i, v_j) = -\frac{1}{C-1}$ for all $i \neq j \in [C]$[1].

[1]This relation for cosines holds if the v_i's are symmetrically distributed such that $\bar{v} = \frac{1}{C}\sum_{i\in[C]} v_i = 0$, and $\cos(v_i, v_j)$ are all the same for $i \neq j$.



(a) Local alignment (b) Local accuracy

Figure 2: Comparison of (a) feature-classifier alignment and (b) accuracy on the observed and unobserved classes test data for θ_r^i trained with $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{DR}}$.



(a) Alignment gap (b) Accuracy gap

Figure 3: Comparison of (a) feature-classifier alignment gap and (b) accuracy gap on the observed and unobserved classes test data for θ_r^i trained with $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{DR}}$.

3 DOT-REGRESSION LOSS MEETS FL

Given our focus on applying $\mathcal{L}_{\text{DR}}$ to FL, we first examine its impact on FL models compared to the CE loss $\mathcal{L}_{\text{CE}}$. In summary, we find that while $\mathcal{L}_{\text{DR}}$ improves alignment and performance on observed class labels, it faces challenge with unobserved classes[2], which are essential for the generalization objective. To address this issue, we propose **FedDr+**, which integrates $\mathcal{L}_{\text{DR}}$ with a novel feature distillation loss. We then evaluate **FedDr+** by analyzing the effect of feature distillation and compare it with various FL algorithms and regularizers.

Experimental configuration. In this section, we conduct experiments on CIFAR-100 [24] using MobileNet [17] with a shard non-iid setting (s=10), where each client contains at most 10 classes. The model is trained for 320 communication rounds, randomly selecting 10% of clients in each round, and the learning rate is decayed at 160^{th} and 240^{th} rounds. The experimental configuration for this section is detailed in subsection 4.1.

3.1 Impact of Dot-Regression Loss on Local and Global Models

We investigate the performance of local models on average when trained with $\mathcal{L}_{\text{DR}}$ compared to $\mathcal{L}_{\text{CE}}$. In Figure 2–3, we calculate the statistics on two datasets: the observed class set O^i, which includes classes present in each client's training data D^i_{train}, and the unobserved class set $\mathcal{U}^i$, consisting of classes unseen during training. This partition highlights the challenges associated with generalizing to unseen classes in FL.

First, we evaluate the feature-classifier alignment $\cos(f(x;\theta_r^i), v_y)$ and accuracy of each local model on the test data (Figure 2). We then observe the amount of change from the given global model to each local model in every communication round (Figure 3). For instance, the alignment gap is denoted by $\cos(f(x;\theta_r^i), v_y) - \cos(f(x;\theta^g_{r-1}), v_y)$.

Performance analysis of local models. Figure 2 shows that $\mathcal{L}_{\text{DR}}$, by focusing its pulling effects exclusively on observed classes within a client's dataset, effectively *enhances alignment and accuracy* for these classes. However, this specificity leads to *poor generalization* on unobserved classes, resulting in significantly weaker performance than models trained with $\mathcal{L}_{\text{CE}}$. Figure 3 displays the different impacts on O^i and $\mathcal{U}^i$ during updates from the global model to local models. $\mathcal{L}_{\text{DR}}$ significantly boosts alignment and accuracy for O^i but causes significant reductions for $\mathcal{U}^i$.

Global model accuracy result. We confirm that $\mathcal{L}_{\text{DR}}$ shows superior accuracy for O^i compared to $\mathcal{L}_{\text{CE}}$ but is less effective at generalizing to $\mathcal{U}^i$. In the shard setting (s = 10)—where each client has access to at most 10 out of 100 classes—this shortcoming significantly reduces the global model's overall accuracy ($\mathcal{L}_{\text{DR}}$: 42.52% vs. $\mathcal{L}_{\text{CE}}$: 46.38%). Thus, it is crucial to develop methods that retain the strengths of $\mathcal{L}_{\text{DR}}$, *i.e.,* alignment of observed classes, while improving generalization for unobserved classes, highlighting the need for more adaptive loss functions in FL.

3.2 FedDr+: Dot-Regression and Feature Distillation for Federated Learning

We propose **FedDr+** to mitigate forgetting unobserved classes while retaining the strengths of dot-regression loss in aligning features of observed classes. Using $\mathcal{L}_{\text{DR}}$ with the frozen classifier $\boldsymbol{V}$, **FedDr+** includes a regularizer that fully distills the global model's feature vectors $f(x;\theta^g) \in \mathbb{R}^d$ to the client features $f(x;\theta)$, to enhance generalization across all classes. The proposed loss function $\mathcal{L}_{\text{Dr+}}$, shown in Equation 1, combines $\mathcal{L}_{\text{DR}}$ with a regularizer $\mathcal{L}_{\text{FD}}(x;\theta,\theta^g) = \frac{1}{d}\|f(x;\theta) - f(x;\theta^g)\|_2^2$. Unless specified, we use a scaling parameter $\beta = 0.9$ throughout the paper.

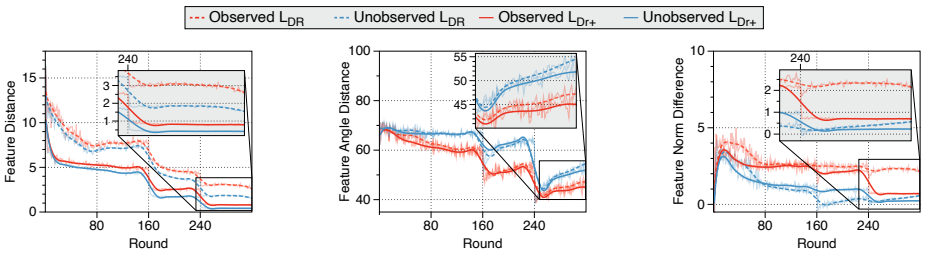
$$\mathcal{L}_{\text{Dr+}}(x,y;\theta,\theta^g,\boldsymbol{V}) = \beta\cdot\mathcal{L}_{\text{DR}}(x,y;\theta,\boldsymbol{V}) + (1-\beta)\cdot\mathcal{L}_{\text{FD}}(x;\theta,\theta^g) \quad (1)$$

Why feature distillation? To address data heterogeneity in FL, various distillation methods have been explored, including model parameters [12, 27, 31, 38], logit-related measurement [4, 19, 25, 27, 34, 40, 47], and co-distillation [5, 6]. In contrast, we utilize the *feature* distillation [15] technique because the feature directly concerns alignment. On the other hand, logits lose information from features when projected onto a frozen ETF classifier [2, 15, 28, 29]. By distilling features, we leverage the global, differentiated knowledge for each data input x. This approach aims to minimize blind

[2]While we use the term "unobserved" in this context, it also applies to "rarely" existing classes.

Table 1: Synergy of various FL algorithms and regularizers. Baseline indicates training FL models without a regularizer. FD denotes feature distillation, which is the regularizer we use in FedDr+.

	Sharding ($s = 10$)						LDA ($\alpha = 0.1$)					
Algorithm	Baseline	+Prox [31]	+KD [16]	+NTD [25]	+LD [23]	+FD	Baseline	+Prox [31]	+KD [16]	+NTD [25]	+LD [23]	+FD
FedAvg [36]	37.22	30.27	35.14	35.56	34.83	37.82	42.52	36.09	41.48	41.34	43.36	43.10
FedBABU [38]	46.20	36.71	45.50	45.09	45.81	45.31	47.37	39.04	45.58	45.56	46.46	44.77
SphereFed [8]	43.90	1.36	41.01	43.47	41.73	45.21	46.98	1.46	45.22	46.25	43.84	48.61
FedETF [33]	32.42	25.18	32.76	31.98	32.25	32.77	46.27	34.92	44.94	45.77	44.36	45.92
Dot-Regression	42.52	5.42	46.60	45.78	47.52	**48.69**	42.72	7.47	48.19	33.08	49.09	**50.79**



(a) Feature distance (b) Feature angle distance (c) Feature norm difference

Figure 4: We present (a) feature distance, (b) feature angle distance, (c) and feature norm difference from θ^g_{r-1} to θ^i_r for observed and unobserved classes by training with $\mathcal{L}_{\text{DR}}$ and $\mathcal{L}_{\text{Dr+}}$.

drift towards observed classes, and hence, we expect it to enhance overall generalization.

3.3 Effect of Feature Distillation

Our findings from subsection 3.1 indicate that $\mathcal{L}_{\text{DR}}$ is unsuitable for the heterogeneous FL environment. This is primarily because there is a notable gap in how features align with the fixed classifier between O^i and $\mathcal{U}^i$. To assess the effect of feature distillation ($\mathcal{L}_{\text{FD}}$), which imposes a constraint on the feature distance $\|f(x;\theta^i_r) - f(x;\theta^g_{r-1})\|_2$ for $x \in O^i$, we measure this distance for both O^i and $\mathcal{U}^i$ from the models trained with $\mathcal{L}_{\text{DR}}$ and $\mathcal{L}_{\text{Dr+}}$. We additionally analyze the angle distance, $\angle(f(x;\theta^i_r), f(x;\theta^g_{r-1}))$, and feature norm difference, $\|f(x;\theta^i_r)\|_2 - \|f(x;\theta^g_{r-1})\|_2$, as these factors influence the feature distance. These values are averaged over the selected client set $\mathcal{S}_r$.

Feature distillation stabilizes the feature dynamics. By adding $\mathcal{L}_{\text{FD}}$, as revealed in Figure 4a, the local model trained with $\mathcal{L}_{\text{Dr+}}$ shows a reduction in feature distance for observed classes, compared to the model trained with $\mathcal{L}_{\text{DR}}$. This reduction happens even for unobserved classes. As demonstrated in Figure 4b and Figure 4c, reduction of feature distance originates from reducing the feature angle distance and feature norm difference for both class sets. In both local models trained with $\mathcal{L}_{\text{DR}}$ and $\mathcal{L}_{\text{Dr+}}$, there is a trend where the angle is significantly larger for $\mathcal{U}^i$ than for O^i (Figure 4b), while the norm difference is smaller for $\mathcal{U}^i$ than for O^i (Figure 4c). This large angle distance of $\mathcal{U}^i$ leads to the degradation of the feature-classifier alignment. By minimizing the angle distance via feature distillation, the global model's accuracy improved substantially, rising from 42.52% with $\mathcal{L}_{\text{DR}}$ to 48.69%with $\mathcal{L}_{\text{Dr+}}$.

Stabilized features enhance alignment and accuracy. We confirm that feature distillation term $\mathcal{L}_{\text{FD}}$ stabilizes feature dynamics for both O^i and $\mathcal{U}^i$, enhancing the global model's capabilities. While the feature difference is stabilized via $\mathcal{L}_{\text{FD}}$, it is essential to verify whether this leads to improved alignment and accuracy.

In Figure 5, we examine in both aspects and illustrate the training curve. Our proposed algorithm, *i.e.,* $\mathcal{L}_{\text{Dr+}}$, demonstrates superior performance for both O^i and $\mathcal{U}^i$ in terms of alignment and accuracy. Notably, even with the addition of a term to the dot-regression loss, alignment is improved. We attribute this improvement to the enhanced knowledge of the global model, which is preserved by preventing the forgetting of previously trained knowledge. Even though the proposed regularizer demonstrates a reasonable regularizing effect, one question remains: "*Is it superior to other previously used regularizers?*"

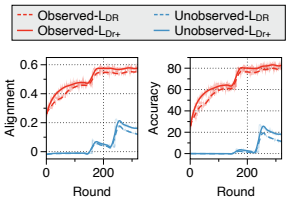


Figure 5: Comparison of alignment/accuracy on the observed and unobserved classes test data for θ^i_r trained with $\mathcal{L}_{\text{DR}}$ and $\mathcal{L}_{\text{Dr+}}$.

3.4 Different FL Algorithms and Regularizers

We answer the above question by evaluating the synergy effect of various FL algorithms by maintaining their original training loss and incorporating specific regularizers, as suggested in Equation 1. Our study includes FedAvg [36] without classifier freezing and other advanced frameworks such as FedBABU [38], SphereFed [8], FedETF [33], and dot-regression, all of which update local models while freezing the classifier. In addition to the FD regularizer, we consider regularizers such as Prox [31] to constrain the distance between local and global model parameters, and several logit-based regularizers—KD [16], NTD [25, 48], and LD [23]—to keep logit-related measurement of local models from deviating significantly from that of the global model. Specifically, KD applies the softened softmax probability from the logit vector, NTD does the same but excludes the true class dimension, and LD distills the entire logit vector. Table 1 demonstrates that **FedDr+** (dot-regression + FD)

Table 2: Accuracy comparison in the GFL setting. The entries are based on results obtained from three different seeds, indicating the mean and standard deviation of the accuracy of the global model, represented as $X_{\pm Y}$. The best performance in each case is highlighted in bold.

	NIID Partition Strategy: Sharding							NIID Partition Strategy: LDA						
	MobileNet on CIFAR-100				VGG on CIFAR-10			MobileNet on CIFAR-100				VGG on CIFAR-10		
Algorithm	s=10	s=20	s=50	s=100	s=2	s=5	s=10	α=0.05	α=0.1	α=0.2	α=0.3	α=0.1	α=0.2	α=0.3
FedAvg [36]	36.63± 0.22	42.25± 1.42	45.57± 0.22	48.20± 1.36	72.08± 0.67	81.53± 0.35	82.38± 0.40	35.58± 1.35	42.10± 0.60	44.78± 0.72	45.73± 0.88	68.71± 1.82	77.75± 0.26	80.76± 0.51
SCAFFOLD [21] (×2)	46.08± 0.37	48.15± 1.21	49.31± 0.62	50.73± 0.42	75.49± 0.42	**84.14**± 0.13	**85.11**± 0.29	40.54± 0.48	46.14± 0.70	47.98± 0.93	48.06± 1.08	*(Failed)*	80.15± 0.29	**82.63**± 0.23
FedNTD [25]	34.05± 1.19	41.78± 0.31	46.42± 0.63	47.17± 0.32	72.21± 0.59	69.96± 17.10	81.99± 0.42	31.78± 3.14	40.41± 0.96	43.10± 2.03	43.04± 0.82	70.22± 0.40	77.16± 0.20	79.50± 0.56
FedExP [20]	36.85± 0.11	42.49± 1.22	45.07± 0.92	48.09± 1.00	72.31± 0.60	81.41± 0.19	82.47± 0.16	34.39± 1.77	40.85± 1.32	44.47± 0.28	45.44± 0.14	70.14± 0.53	78.09± 0.21	80.40± 0.54
FedBABU [38]	45.97± 0.48	45.53± 0.79	46.52± 0.51	46.02± 0.28	71.99± 0.52	81.07± 0.60	82.32± 0.06	41.97± 1.01	45.77± 0.28	44.28± 0.45	44.80± 0.63	65.15± 3.66	77.03± 0.25	79.91± 0.13
SphereFed [8]	42.71± 0.65	48.63± 0.90	**52.16**± 0.22	**53.41**± 0.19	**76.33**± 0.33	83.67± 0.18	84.36± 0.30	39.56± 0.48	46.54± 0.58	**49.41**± 0.78	49.22± 0.86	67.49± 3.49	80.05± 0.40	**82.62**± 0.66
FedETF [33]	31.37± 0.72	42.22± 0.77	47.47± 0.67	49.00± 0.74	67.81± 0.94	80.78± 0.68	82.60± 0.46	40.71± 0.90	45.63± 0.33	46.28± 1.05	46.69± 0.87	70.75± 0.36	77.86± 0.46	79.95± 0.34
FedDr+ (Ours)	**48.21**± 0.56	**50.77**± 0.14	**52.15**± 0.03	**52.41**± 0.81	**76.57**± 0.51	83.22± 0.34	84.14± 0.27	**45.12**± 1.00	**49.48**± 0.50	**50.67**± 0.88	**51.15**± 0.65	**72.07**± 2.26	**80.90**± 0.02	**82.42**± 0.10

achieves the best performance. Generally, Prox tends to be less effective than logit-based regularizers, which are often outperformed by FD across most algorithms. This is because, as noted in subsection 3.2, with the frozen classifier, features are expected to have rich information to mitigate the drift. Prox uniformly regularizes all data instances, whereas logit and feature regularizers adapt to both model parameters and data instances, offering more refined control. Specifically, FD regularizer, with its higher dimensionality, captures the global model's information more precisely than logit-based ones, resulting in better synergy.

4 EXPERIMENTS AND RESULTS

In this section, we present the experimental results of **FedDr+**, specifically focusing on global federated learning (GFL). Furthermore, a comprehensive evaluation of **FedDr+**, including personalized federated learning (PFL) results, analyses of hyper-parameters such as local epochs, client sampling ratio, the effect of different β values in **FedDr+**, and the elapsed time results, is detailed in Appendix D.

4.1 Experimental Setup

Dataset and models. To simulate a realistic FL scenario involving 100 clients, we conduct extensive studies on two widely used datasets: CIFAR-10 and CIFAR-100 [24]. For CIFAR-10, we employ VGG11 [42], while for CIFAR-100, MobileNet [17] is used. The training data is distributed among 100 clients using sharding and the LDA (Latent Dirichlet Allocation) partition strategies. Following the convention, sharding distributes the data into non-overlapping shards of equal size, each shard encompassing $\frac{|D_{\text{train}}|}{100\times s}$ and $\frac{|D_{\text{test}}|}{100\times s}$ samples per class, where s denotes the number of shards per client. On the other hand, LDA involves sampling a probability vector from Dirichlet distribution, $p_c = (p_{c,1}, p_{c,2}, \cdots, p_{c,100}) \sim \text{Dir}(\alpha)$, and allocating a proportion $p_{c,k}$ of instances of class $c \in [C]$ to each client $k \in [100]$. Smaller values of s and α increase the level of data heterogeneity.

Implementation details. In each round of communication, a fraction of clients equal to 0.1 is randomly selected to participate in the training process. The total number of communication rounds is 320. The initial learning rate and the number of local epochs for CIFAR-10 and CIFAR-100 are determined through grid searches, with the detailed process and results provided in Appendix C. The learning rate η is decayed by a factor of 0.1 at the 160th and 240th communication rounds. The number of local epochs is set to 10 for CIFAR-10 and 3 for CIFAR-100 in the main experiments.

4.2 Global Federated Learning Results

We compare **FedDr+** with a range of GFL algorithms, considering both non-freezing and freezing classifier approaches. Among non-freezing classifiers, **FedDr+** competes with FedAvg [36], SCAFFOLD [21], FedNTD [25], and FedExP [20]. **FedDr+** is also evaluated against freezing classifier algorithms such as FedBABU [38], SphereFed [8], and FedETF [33]. Among the baseline algorithms, SCAFFOLD incurs a communication cost two times higher per round, denoted as (×2). Our experiments encompass heterogeneous settings involving sharding and LDA non-IID environments.

Table 2 summarizes the accuracy comparison between the state-of-the-art GFL methods and FedAvg under various conditions. While specific methods demonstrated effectiveness in particular scenarios, some of these frequently underperformed relative to the robustness of FedAvg. For example, SCAFFOLD shown strong performance in the less heterogeneous sharding setting on CIFAR-10; however, it failed in model training under the highly heterogeneous LDA condition with $\alpha = 0.1$. Notably, **FedDr+** consistently exceeded all baseline methods in performance across diverse experimental conditions and often achieved state-of-the-art results. **FedDr+** demonstrated exceptional performance in highly heterogeneous FL environments, particularly excelling in the CIFAR-100 LDA configuration with $\alpha = 0.05$, achieving a notable 3.15% improvement over all baseline models.

5 CONCLUSION

Motivated by the recent FL methods enhancing feature alignment with a fixed classifier, we first investigate the effects of applying dot-regression loss for FL. Since the dot-regression is the most direct method for feature-classifier alignment, we find it improves alignment and accuracy in local models but degrades the performance of the global model. This happens because local clients trained with dot-regression tend to forget classes that have not been observed. To address this, we propose **FedDr+**, combining dot-regression with a feature distillation method. By regularizing the deviation of local features from global features, **FedDr+** allows local models to maintain knowledge about all classes during training, thereby ultimately preserving general knowledge of the global model. Our method achieves top performance in FL experiments, even when data is distributed unevenly across devices (non-IID settings).

ACKNOWLEDGEMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by Korea government (MSIT) [No. 2021-0-00907, Development of Adaptive and Lightweight Edge-Collaborative Analysis Technology for Enabling Proactively Immediate Response and Rapid Learning, 90%] and [No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST), 10%].

REFERENCES

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818* (2019).

[2] Emanuel Ben-Baruch, Matan Karklinsky, Yossi Biton, Avi Ben-Cohen, Hussam Lawen, and Nadav Zamir. 2022. It's all in the head: Representation knowledge distillation through classifier sharing. *arXiv preprint arXiv:2201.06945* (2022).

[3] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. 2023. On the Importance and Applicability of Pre-Training for Federated Learning. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=fWWFv--P0xP

[4] Wei-Chun Chen, Chia-Che Chang, and Che-Rung Lee. 2019. Knowledge distillation with feature maps for image classification. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer, 200–215.

[5] Zihan Chen, Howard Yang, Tony Quek, and Kai Fong Ernest Chong. 2024. Spectral Co-Distillation for Personalized Federated Learning. *Advances in Neural Information Processing Systems* 36 (2024).

[6] Yae Jee Cho, Jianyu Wang, Tarun Chirvolu, and Gauri Joshi. 2023. Communication-efficient and model-heterogeneous personalized federated learning via clustered knowledge transfer. *IEEE Journal of Selected Topics in Signal Processing* 17, 1 (2023), 234–247.

[7] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*. PMLR, 2089–2099.

[8] Xin Dong, Sai Qian Zhang, Ang Li, and HT Kung. 2022. Spherefed: Hyperspherical federated learning. In *European Conference on Computer Vision*. Springer, 165–184.

[9] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems* 33 (2020), 3557–3568.

[10] Ziqing Fan, Jiangchao Yao, Bo Han, Ya Zhang, Yanfeng Wang, et al. 2024. Federated Learning with Bilateral Curation for Partially Class-Disjoint Data. *Advances in Neural Information Processing Systems* 36 (2024).

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[12] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems* 33 (2020), 14068–14080.

[13] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[15] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. 2019. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1921–1930.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[18] Chenxi Huang, Liang Xie, Yibo Yang, Wenxiao Wang, Binbin Lin, and Deng Cai. 2023. Neural Collapse Inspired Federated Learning with Non-iid Data. arXiv:2303.16066 [cs.LG]

[19] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. 2021. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing* 22, 1 (2021), 191–205.

[20] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. 2023. FedExP: Speeding Up Federated Averaging via Extrapolation. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=IPrzNbddXV

[21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.

[22] Seongyoon Kim, Gihun Lee, Jaehoon Oh, and Se-Young Yun. 2023. FedFN: Feature Normalization for Alleviating Data Heterogeneity Problem in Federated Learning. *arXiv preprint arXiv:2311.13267* (2023).

[23] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. 2021. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. *arXiv preprint arXiv:2105.08919* (2021).

[24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html* 6, 1 (2009), 1.

[25] Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. 2022. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 38461–38474.

[26] José Lezama, Qiang Qiu, Pablo Musé, and Guillermo Sapiro. 2018. Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8109–8118.

[27] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[28] Jingzhi Li, Zidong Guo, Hui Li, Seungju Han, Ji-won Baek, Min Yang, Ran Yang, and Sungjoo Suh. 2023. Rethinking feature-based knowledge distillation for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20156–20165.

[29] Quanquan Li, Shengying Jin, and Junjie Yan. 2017. Mimicking very efficient network for object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition*. 6356–6364.

[30] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*. PMLR, 6357–6368.

[31] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[32] Xin-Chun Li and De-Chuan Zhan. 2021. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 995–1005.

[33] Zexi Li, Xinyi Shang, Rui He, Tao Lin, and Chao Wu. 2023. No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5319–5329.

[34] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 2351–2363.

[35] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. 2021. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems* 34 (2021), 5972–5984.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[37] John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. 2022. Where to begin? on the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2206.15387* (2022).

[38] Jaehoon Oh, SangMook Kim, and Se-Young Yun. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HuaYQfggn5u

[39] Vardan Papyan, XY Han, and David L Donoho. 2020. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences* 117, 40 (2020), 24652–24663.

[40] Biao Qian, Yang Wang, Hongzhi Yin, Richang Hong, and Meng Wang. 2022. Switchable online knowledge distillation. In *European Conference on Computer Vision*. Springer, 449–466.

[41] Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* (2013).

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BkluqlSFDS

[44] Zikai Xiao, Zihan Chen, Liyinglan Liu, YANG FENG, Joey Tianyi Zhou, Jian Wu, Wanlu Liu, Howard Hao Yang, and Zuozhu Liu. 2024. FedLoGe: Joint Local and Generic Federated Learning under Long-tailed Data. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=V3j5d0GQgH

[45] Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. 2022. Inducing Neural Collapse in Imbalanced Learning: Do We Really Need a Learnable Classifier at the End of Deep Neural Network? *Advances in Neural Information Processing Systems* 35 (2022), 37991–38002.

[46] Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. 2023. Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. *arXiv preprint arXiv:2302.03004* (2023).

[47] Rui Ye, Yaxin Du, Zhenyang Ni, Yanfeng Wang, and Siheng Chen. 2024. Fake It Till Make It: Federated Learning with Consensus-Oriented Generation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=NY3wMJuaLf

[48] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. 2022. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 11953–11962.

- Appendix -

FedDr+: Stabilizing Dot-regression with Global Feature Distillation for Federated Learning

We organized notations at Appendix A. In Appendix B, we show the pulling and pushing gradients of the CE loss in detail. Then, we elucidate the experimental setup in Appendix C, encompassing dataset description, model specifications, NIID partition, and hyperparameter search. In Appendix D, We present additional experimental results of PFL, sensitivity analysis, and elapsed time measurement.

A NOTATIONS

Table 3: Notations used throughout the paper.

Indices	
$c \in [C]$	Index for a class
$r \in [R]$	Index for FL round
$i \in [N]$	Index for a client
Dataset	
D^i_{train}	Training dataset for client i
D^i_{test}	Test dataset for client i
$(x, y) \in D^i_{\text{train,test}}\,; (x, y) \sim \mathcal{D}^i$	Data on client i sampled from distribution $\mathcal{D}^i$ (x: input data, y: class label)
$\mathcal{O}^i$	Dataset consists of observed classes in client i
$\mathcal{U}^i$	Dataset consists of unobserved classes in client i
Parameters	
$\boldsymbol{\theta}$	Feature extractor weight parameters
$\boldsymbol{V} = [v_1, \ldots, v_C] \in \mathbb{R}^{C\times d}$	Classifier weight parameters (frozen during training)
$v_c, c \in [C]$	c-th row vector of $\boldsymbol{V}$
$\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{V})$	All model parameters
$\boldsymbol{\Theta}^g_r = (\boldsymbol{\theta}^g_r, \boldsymbol{V})$	Aggregated global model parameters at round r
$\boldsymbol{\Theta}^i_r = (\boldsymbol{\theta}^i_r, \boldsymbol{V})$	Trained model parameters on client i at round r
Model Forward	
$p(x; \boldsymbol{\theta}) \in \mathbb{R}^C$	Softmax probability of input x
$p_c(x; \boldsymbol{\theta}), c \in [C]$	c-th element of $p(x; \boldsymbol{\theta})$
$\mathcal{L}_{\text{CE}}(x; \theta) = -\log p_y(x; \boldsymbol{\theta})$	Cross-entropy loss of input x
$f(x; \boldsymbol{\theta}) \in \mathbb{R}^d$	Feature vector of input x
$z(x; \boldsymbol{\theta}) = f(x; \boldsymbol{\theta})\boldsymbol{V}^\top \in \mathbb{R}^C$	Logit vector of input x
$z_c(x; \boldsymbol{\theta}), c \in [C]$	c-th element of $z(x; \boldsymbol{\theta})$

B PRELIMINARIES: PULLING AND PUSHING FEATURE GRADIENTS IN CE

In this section, we first calculate the classifier gradient for features and introduce the pulling and pushing effects of the cross-entropy objective.

B.1 Feature Gradient of $\mathcal{L}_{\text{CE}}$

We first provide two lemmas supporting Proposition 1, explaining the behavior of pulling and pushing feature gradients in the cross-entropy (CE) loss.

Lemma 1. *For all* $c, c' \in [C]$, $\frac{\partial p_{c'}(x;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})} = \begin{cases} p_c(x;\boldsymbol{\theta}) \cdot (1 - p_c(x;\boldsymbol{\theta})) & \text{if } c = c' \\ -p_c(x;\boldsymbol{\theta}) \cdot p_{c'}(x;\boldsymbol{\theta}) & \text{else} \end{cases}$.

Proof. Note that $p(x;\boldsymbol{\theta}) = \left[\frac{\exp(z_j(x;\boldsymbol{\theta}))}{\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))}\right]_{j=1}^C \in \mathbb{R}^C$. Then,

(i) $c = c'$ case:

$$\begin{aligned}\frac{\partial p_c(x;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})} &= \frac{\partial}{\partial z_c(x;\boldsymbol{\theta})}\left\{\frac{\exp(z_c(x;\boldsymbol{\theta}))}{\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))}\right\} \\ &= \frac{\exp(z_c(x;\boldsymbol{\theta}))\left(\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))\right) - \exp(z_c(x;\boldsymbol{\theta}))^2}{\left(\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))\right)^2} \\ &= p_c(x;\boldsymbol{\theta}) - p_c(x;\boldsymbol{\theta})^2 = p_c(x;\boldsymbol{\theta})(1 - p_c(x;\boldsymbol{\theta})).\end{aligned}$$

(ii) $c \neq c'$ case:

$$\begin{aligned}\frac{\partial p_{c'}(x;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})} = \frac{\partial}{\partial z_c(x;\boldsymbol{\theta})}\left\{\frac{\exp(z_{c'}(x;\boldsymbol{\theta}))}{\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))}\right\} &= \frac{-\exp(z_c(x;\boldsymbol{\theta}))\exp(z_{c'}(x;\boldsymbol{\theta}))}{\left(\sum_{i=1}^C \exp(z_i(x;\boldsymbol{\theta}))\right)^2} \\ = -p_c(x;\boldsymbol{\theta})p_{c'}(x;\boldsymbol{\theta}).\end{aligned}$$

□

Lemma 2. $\nabla_{z(x;\boldsymbol{\theta})}\mathcal{L}_{CE}(x, y;\boldsymbol{\theta}) = p(x;\boldsymbol{\theta}) - \mathbf{e}_y$, *where* $\mathbf{e}_y \in \mathbb{R}^C$ *is the unit vector with its* y*-th element as 1.*

Proof.

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{CE}}(x, y;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})} = -\frac{\partial}{\partial z_c(x;\boldsymbol{\theta})}\log p_y(x;\boldsymbol{\theta}) &= -\frac{1}{p_y(x;\boldsymbol{\theta})}\frac{\partial p_y(x;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})} \\ \overset{(\bigstar)}{=} \begin{cases} p_c(x;\boldsymbol{\theta}) - 1 & \text{if } c = y \\ p_c(x;\boldsymbol{\theta}) & \text{else} \end{cases} &= p_c(x;\boldsymbol{\theta}) - \mathbb{1}\{c = y\}.\end{aligned}$$

Note that ($\bigstar$) holds by the Lemma 1. Therefore, the desired result is satisfied. □

Proposition 1. *Given* (x, y)*, the gradient of the* $\mathcal{L}_{CE}$ *with respect to* $f(x;\boldsymbol{\theta})$ *is given by:*

$$\nabla_{f(x;\boldsymbol{\theta})}\mathcal{L}_{CE}(x, y;\boldsymbol{\theta}) = -(1 - p_y(x;\boldsymbol{\theta}))v_y + \sum_{c\in[C]\setminus\{y\}} p_c(x;\boldsymbol{\theta})v_c. \tag{2}$$

Proof.

$$
\begin{aligned}
&\nabla_{f(x;\boldsymbol{\theta})}\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta}) \\
&\overset{(\clubsuit)}{=} \left[\nabla_{f(x;\boldsymbol{\theta})}z_1(x;\boldsymbol{\theta})|\cdots|\nabla_{f(x;\boldsymbol{\theta})}z_C(x;\boldsymbol{\theta})\right]\nabla_{z(x;\boldsymbol{\theta})}\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta}) \\
&= \sum_{c=1}^{C}\frac{\partial\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})}\nabla_{f(x;\boldsymbol{\theta})}z_c(x;\boldsymbol{\theta}) \\
&= \frac{\partial\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})}{\partial z_y(x;\boldsymbol{\theta})}\nabla_{f(x;\boldsymbol{\theta})}z_y(x;\boldsymbol{\theta}) + \sum_{c\in[C]\setminus\{y\}}\frac{\partial\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})}\nabla_{f(x;\boldsymbol{\theta})}z_c(x;\boldsymbol{\theta}) \\
&= \frac{\partial\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})}{\partial z_y(x;\boldsymbol{\theta})}v_y + \sum_{c\in[C]\setminus\{y\}}\frac{\partial\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})}{\partial z_c(x;\boldsymbol{\theta})}v_c \\
&\overset{(\spadesuit)}{=} -(1-p_y(x;\boldsymbol{\theta}))v_y + \sum_{c\in[C]\setminus\{y\}}p_c(x;\boldsymbol{\theta})v_c.
\end{aligned}
$$

Employing the chain rule for (♣) and invoking Lemma 2 for (♠) confirms the result.

□

B.2 Physical Meaning of $\nabla_{f(x;\theta)}\mathcal{L}_{\mathbf{CE}}(x,y;\theta)$

Note that $\nabla_{f(x;\boldsymbol{\theta})}\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})$ has two components: $\mathbf{F}_{\mathrm{Pull}} = (1-p_y(x;\boldsymbol{\theta}))v_y$ and $\mathbf{F}_{\mathrm{Push}} = -\sum_{c\in[C]\setminus\{y\}}p_c(x;\boldsymbol{\theta})v_c$. $\mathbf{F}_{\mathrm{Pull}}$ adjusts the feature vector in the positive direction of the actual class index's classifier vector v_y, guiding alignment towards v_y. Conversely, $\mathbf{F}_{\mathrm{Push}}$ adjusts the feature vector in the negative direction of the vectors in the not-true class set $[C]\setminus\{y\}$, inducing misalignment towards v_c for $c\in[C]\setminus\{y\}$.

C EXPERIMENTAL SETUP

C.1 Code Implementation

Our implementations are conducted using the PyTorch framework. Specifically, the experiments presented in Table 2 are executed on a single NVIDIA RTX 3090 GPU, based on the code structure from the following repository: https://github.com/Lee-Gihun/FedNTD. The other parts of our study are carried out on a single NVIDIA A5000 GPU, utilizing the code framework from https://github.com/jhoon-oh/FedBABU.

C.2 Datasets, Model, and Optimizer

To simulate a realistic FL scenario, we conduct extensive studies on two widely used datasets: CIFAR-10 and CIFAR-100 [24]. A momentum optimizer is utilized for all experiments. Unless otherwise noted, the basic setting of our experiments follows the dataset statistics, FL scenario specifications, and optimizer hyperparameters summarized in Table 4.

Table 4: Summary of Dataset, Model, FL System, and Optimizer Specifications

Datasets	C	$\lvert D_{\mathrm{train}}\rvert$	$\lvert D_{\mathrm{test}}\rvert$	N	R	r	E	B	m	λ
CIFAR-10	10	50000	10000	100	320	0.1	10	50	0.9	1e-5
CIFAR-100	100	50000	10000	100	320	0.1	3	50	0.9	1e-5

Note: In terms of dataset information, C represents the number of classes in the dataset, with $\lvert D_{\mathrm{train}}\rvert$ and $\lvert D_{\mathrm{test}}\rvert$ indicating the total numbers of training and test data used, respectively. For the federated learning (FL) system specifics, R indicates the total number of FL rounds, r is the ratio of clients selected for each round, and E denotes the number of local epochs. Local model training utilizes a momentum optimizer where B is the batch size, and m and λ represent the momentum and weight decay parameters, respectively. The initial learning rate η is decayed by a factor of 0.1 at the 160th and 240th communication rounds. The initial learning rate η and batch size B were determined via extensive grid search for each algorithm, details outlined in Appendix C.4.

C.3 Non-IID Partition Strategies

To induce heterogeneity in each client's training and test data $(D^i_{\mathrm{train}}, D^i_{\mathrm{test}})$, we distribute the entire class-balanced datasets, D_{train} and D_{test}, among 100 clients using both sharding and Latent Dirichlet Allocation (LDA) partitioning strategies:

- **Sharding** [36, 38]: We organize the D_{train} and D_{test} by label and divide them into non-overlapping shards of equal size. Each shard encompasses $\frac{\lvert D_{\mathrm{train}}\rvert}{100\times s}$ and $\frac{\lvert D_{\mathrm{test}}\rvert}{100\times s}$ samples of the same class, where s denotes the number of shards per client. This sharding technique is

Table 5: Hyperparameters for VGG11 training on CIFAR-10.

	Feature un-normalized algorithms					Feature normalized algorithms		
Hyperparameters	FedAvg	FedBABU	SCAFFOLD	FedNTD	FedExP	FedETF	SphereFed	**FedDr+** (Ours)
η	0.01	0.01	0.01	0.01	0.01	0.05	0.55	0.35
Additional	None	None	None	(β, τ)=(1,3)	ϵ=0.001	(β, τ)=(1,1)	None	β=0.9

Table 6: Hyperparameters for MobileNet training on CIFAR-100.

	Feature un-normalized algorithms					Feature normalized algorithms		
Hyperparameters	FedAvg	FedBABU	SCAFFOLD	FedNTD	FedExP	FedETF	SphereFed	**FedDr+** (Ours)
η	0.1	0.1	0.1	0.1	0.1	0.5	6.5	5.0
Additional	None	None	None	(β, τ)=(1,3)	ϵ=0.001	(β, τ)=(1,1)	None	β=0.9

used to create D^i_{train} and D^i_{test}, which are then distributed to each client i, ensuring that each client has the same number of training and test samples. The data for each client is disjoint. As a result, each client has access to a maximum of s different classes. Decreasing the number of shards per user s increases the level of data heterogeneity among clients.

- **Latent Dirichlet Allocation (LDA)** [35, 43]: We utilize the LDA technique to create D^i_{train} from D_{train}. This involves sampling a probability vector $p_c = (p_{c,1}, p_{c,2}, \cdots, p_{c,100}) \sim \text{Dir}(\alpha)$ and allocating a proportion $p_{c,k}$ of instances of class $c \in [C]$ to each client $k \in [100]$. Here, $Dir(\alpha)$ represents the Dirichlet distribution with the concentration parameter α. The parameter α controls the strength of data heterogeneity, with smaller values leading to stronger heterogeneity among clients. For D^i_{test}, we randomly sample from D_{test} to match the class frequency of D^i_{train} and distribute it to each client i.

C.4 Hyperparameter Search for η and E

To optimize the initial learning rate (η) and the number of local epochs (E) for our algorithm, we conduct a grid search on the CIFAR-10 and CIFAR-100 datasets. The process and reasoning are outlined below.

Rationale for varying initial learning rate (η). The algorithms used in our experiments differ in handling feature normalization within the loss function. Some algorithms apply feature normalization, while others do not. When features $f(x;\boldsymbol{\theta})$ are normalized, the resulting gradient is scaled by $\frac{1}{\|f(x;\boldsymbol{\theta})\|_2}$. This scaling effect necessitates a grid search across various learning rates to account for the differences in learning behavior.

Rationale for varying local epochs (E). In FL, choosing the appropriate number of local epochs is crucial. Too few epochs can lead to underfitting, while too many can cause client drift. Therefore, finding the optimal number of local epochs is essential by exploring a range of values.

Grid search process and results. Considering the above reasons, we perform grid search for η and E on CIFAR-10 and CIFAR-100 datasets. The grid search for CIFAR-10 uses a shard size of 2, while for CIFAR-100, a shard size of 10 is used. The detailed procedures for each dataset are provided below. These optimal settings have also been confirmed to yield good performance in less heterogeneous settings.

- **CIFAR-10**: We examine η values from {0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6}. For E, we consider {1, 3, 5, 10, 15}. The optimal learning rates vary by algorithm, and the results are summarized in Table 5. Table 5 also includes the additional hyperparameters used for each algorithm. The notation for these additional hyperparameters follows the conventions used throughout this paper. The optimal number of local epochs is found to be 10 for every algorithm.
- **CIFAR-100**: We examine η values from {0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0}. A default initial learning rate of 0.1 is used unless specified otherwise. The optimal learning rates differ by algorithm, and the results are listed in Table 6. Table 6 also includes the additional hyperparameters used for each algorithm. The notation for these additional hyperparameters follows the conventions used throughout this paper. The optimal number of local epochs is found to be 3 for every algorithm.

D ADDITIONAL EXPERIMENT RESULTS

D.1 Personalized Federated Learning Results

Table 7: PFL accuracy comparison with MobileNet on CIFAR-100. For PFL, we denote the entries in the form of $X_{\pm(Y)}$, representing the mean and standard deviation of personalized accuracies across all clients derived from a single seed.

Algorithm	s=10	s=20	s=100	α=0.05	α=0.1	α=0.3
Local only ($\mathcal{L}_{\text{CE}}$)	58.05±(8.11)	42.45±(6.44)	18.69±(3.28)	55.39±(8.79)	43.76±(7.46)	27.75±(5.32)
Local only ($\mathcal{L}_{\text{CE}}$+ETF)	58.01±(7.34)	41.62±(5.91)	18.92±(3.00)	55.34±(9.13)	43.37±(7.12)	27.87±(5.34)
Local only ($\mathcal{L}_{\text{DR}}$)	60.68±(7.77)	44.61±(6.61)	20.98±(3.49)	58.56±(9.16)	46.72±(7.29)	30.88±(5.33)
FedPer [1]	70.67±(7.19)	57.27±6.66	24.30±(4.34)	62.67±(7.65)	53.43±(6.60)	35.68±(4.82)
Per-FedAvg [9]	32.13±(10.90)	36.66±(8.86)	41.27±(7.43)	28.81±(8.68)	35.56±(6.56)	42.80±(4.76)
FedRep [7]	63.14±(7.63)	51.69±(6.50)	26.31±(4.74)	57.53±(8.05)	49.60±(6.25)	37.00±(4.82)
Ditto [30]	39.26±(14.43)	38.18±(9.96)	44.53±(5.08)	35.81±(14.83)	37.81±(11.80)	43.72±(5.12)
FedAvg-FT [36]	69.81±(6.78)	56.13±(5.77)	47.66±(5.20)	63.37±(9.28)	56.79±(5.96)	50.12±(3.67)
FedBABU-FT [38]	80.14±(6.25)	70.89±(5.60)	52.14±(5.09)	75.50±(6.40)	70.83±(5.06)	56.91±(3.74)
SphereFed-FT [8]	81.90±(5.86)	71.56±(5.78)	55.83±(4.67)	73.21±(7.08)	70.00±(5.09)	60.03±(3.99)
FedETF-FT [33]	53.75±(7.35)	52.94±(5.71)	51.69±(5.03)	52.96±(8.01)	53.97±(5.40)	51.67±(3.83)
FedDr+ FT (ours)	**84.10**±(5.43)	**75.42**±(4.80)	**56.76**±(4.91)	**78.55**±(6.16)	**74.75**±(4.75)	**62.16**±(3.73)

We introduce **FedDr+** FT, inspired by prior work [8, 22, 33, 38], which enhances personalization by leveraging local data to fine-tune the global federated learning (GFL) model. We fine-tune the **FedDr+** GFL model using $\mathcal{L}_{\text{Dr+}}$ to create **FedDr+** FT, *i.e.,* 2-step approach. For a comprehensive analysis, we compare **FedDr+** FT with existing personalized federated learning (PFL) methods, including 1-step approaches, *i.e.,* creating PFL models from scratch, such as FedPer [1], Per-FedAvg [9], FedRep [7], and Ditto [30], as well as 2-step methods such as FedAVG-FT, FedBABU-FT [38], SphereFed-FT [8], and FedETF-FT [33]. Additionally, we compare these methods with various simple local models that have not undergone federated learning: (1) Local only ($\mathcal{L}_{\text{CE}}$), trained with $\mathcal{L}_{\text{CE}}$, (2) Local only ($\mathcal{L}_{\text{CE}}$ + ETF), trained with $\mathcal{L}_{\text{CE}}$ and initializing the classifier with an ETF classifier, and (3) Local only ($\mathcal{L}_{\text{DR}}$), trained using $\mathcal{L}_{\text{DR}}$.

In Table 7, we first compare the performance of simple local models in PFL by examining $\mathcal{L}_{\text{DR}}$ and $\mathcal{L}_{\text{CE}}$. While methods using $\mathcal{L}_{\text{CE}}$ show no significant differences, utilizing $\mathcal{L}_{\text{DR}}$ leads to substantial performance improvements in PFL across all settings. The "Local only ($\mathcal{L}_{\text{CE}}$)" and "Local only ($\mathcal{L}_{\text{CE}}$ + ETF)" methods exhibit similar performance due to the nearly classwise orthogonal nature of randomly initialized classifiers [11, 14, 26, 38, 41]. With a large number of classes (C=100), the ETF classifier, which is also nearly classwise orthogonal, performs similarly to random initialization. When comparing **FedDr+** FT with other 2-step methods, **FedDr+** FT consistently demonstrates superior performance. This aligns with previous research [3, 37] suggesting that fine-tuning from a well-initialized model yields better PFL performance. Additionally, compared with 1-step algorithms, **FedDr+** FT continues to show superiority, outperforming all baseline methods across all settings.

D.2 Evaluating Dot-Regression and FedDr+ for Personalized Federated Learning

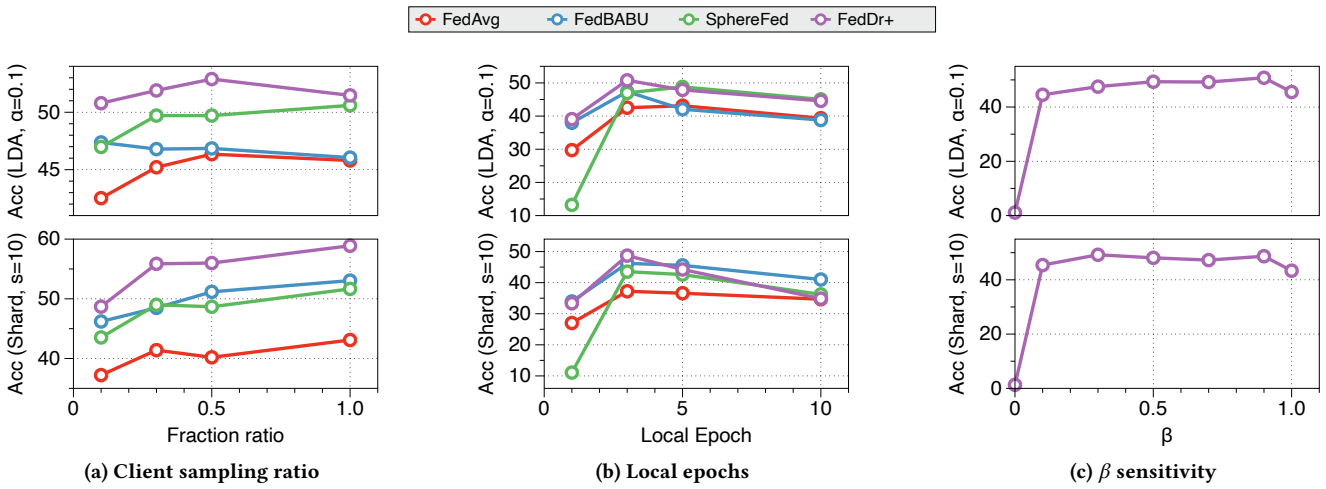
Table 8: PFL accuracy comparison based on dot-regression and FedDr+ with MobileNet on CIFAR-100. For PFL, we denote the entries in the form of $X_{\pm(Y)}$, representing the mean and standard deviation of personalized accuracies across all clients, derived from a single seed.

Algorithm	s=10	s=20	s=100	α=0.05	α=0.1	α=0.3
Dot-Regression	42.52	49.02	52.86	30.31±7.95	37.52±5.60	47.08±3.69
Dot-Regression FT ($\mathcal{L}_{\text{DR}}$)	80.84±(5.99)	74.18±(5.78)	56.84±(5.04)	72.02±(6.80)	66.96±(5.36)	60.34±(3.66)
Dot-Regression FT ($\mathcal{L}_{\text{Dr+}}$)	80.82±(6.12)	73.73±(5.75)	56.69±(4.95)	71.85±(7.03)	66.59±(5.32)	59.87±(3.65)
FedDr+ (ours)	48.69	51.00	53.23	39.63±9.12	45.83±6.18	48.04±3.44
FedDr+ FT ($\mathcal{L}_{\text{DR}}$) (ours)	**84.23**±(5.44)	**75.73**±(4.79)	**56.90**±(4.85)	**78.65**±(6.17)	**74.86**±(4.77)	**62.47**±(3.72)
FedDr+ FT ($\mathcal{L}_{\text{Dr+}}$) (ours)	84.10±(5.43)	75.42±(4.80)	56.76±(4.91)	78.55±(6.16)	74.75±(4.75)	62.16±(3.73)

We introduce **FedDr+** FT and dot-regression FT, inspired by prior work [8, 22, 33, 38]. These methods enhance personalization by leveraging local data to fine-tune the GFL model. We investigate the impact of fine-tuning using $\mathcal{L}_{\text{Dr+}}$ and $\mathcal{L}_{\text{DR}}$ loss for each GFL model to assess their effectiveness on personalized accuracy. Performance metrics without standard deviations indicate results on D_{test}, obtained from the GFL model after the initial step in the 2-step method. Our experiments involve heterogeneous settings with sharding and LDA non-IID environments, using MobileNet on CIFAR-100 datasets. We set s as 10, 20, and 100, and the LDA concentration parameter (α) as 0.05, 0.1, and 0.3. Table 8 provides detailed personalized accuracy results.

Our 2-step process involves first developing the GFL model either using dot-regression or **FedDr+**. In the second step, we fine-tune this model to create the PFL model, again using $\mathcal{L}_{\text{DR}}$ or $\mathcal{L}_{\text{Dr+}}$. This results in four combinations: Dot-Regression FT ($\mathcal{L}_{\text{DR}}$), Dot-Regression FT ($\mathcal{L}_{\text{Dr+}}$), **FedDr+** FT ($\mathcal{L}_{\text{DR}}$), and **FedDr+** FT ($\mathcal{L}_{\text{Dr+}}$). When the GFL model is fixed, using $\mathcal{L}_{\text{DR}}$ for fine-tuning consistently outperforms $\mathcal{L}_{\text{Dr+}}$ across all settings, because dot-regression focuses on local alignment which advantages personalized fine-tuning. Conversely, when the fine-tuning method is fixed, employing $\mathcal{L}_{\text{Dr+}}$ for the GFL model consistently outperforms $\mathcal{L}_{\text{DR}}$ across all settings. This aligns with previous research [3, 37] suggesting that fine-tuning from a well-initialized model yields better PFL performance.

D.3 Sensitivity Analysis



(a) Client sampling ratio **(b) Local epochs** **(c) β sensitivity**

Figure 6: Performance of baselines and FedDr+ on CIFAR-100 (α=0.1 and s=10) with various analyses: (a) client sampling ratio, (b) the number of local epochs, and (c) sensitivity to β.

We explore the impact of varying client sampling ratio and local epochs on performance, as well as the effect of different β values in **FedDr+**, as detailed in Figure 6. All experiments are conducted on MobileNet using the CIFAR-100 dataset with sharding (s=10) and LDA (α=0.1).

Effect of client sampling ratio and local epochs. We evaluate the sensitivity of hyperparameters in **FedDr+** by comparing it to baselines under varying client sampling ratio and local epochs, starting from the default setting of client sampling ratio of 0.1 and local epoch of 3. Compared to FedAvg (without classifier freezing), FedBABU and SphereFed (all with classifier freezing) show performance improvements with increasing fraction ratios, but **FedDr+** consistently outperforms the baselines. The number of local epochs is crucial in FL; too few epochs result in underfitting, while too many cause client drift, degrading global model performance. The default setting of local epochs 3 is optimal for all baselines, with **FedDr+** achieving the best performance. Although performance generally declines when deviating from this peak point, **FedDr+** remains the best or highly competitive.

Weight ratio β analysis. We analyze the effect of scaling parameter in **FedDr+** by varying β while keeping other hyperparameters constant. The performance is evaluated for $\beta \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. When $\beta = 0$, only feature distillation is applied, and when $\beta = 1$, only dot-regression is used. $\beta \in \{0, 1\}$ are generally less effective, whereas $\beta \in \{0.3, 0.5, 0.7, 0.9\}$ show consistently good performance, indicating a balanced approach is beneficial.

D.4 Elapsed Time Results

Table 9: Elapsed time per round (in seconds) for various GFL algorithms.

	Non-feature normalized algorithms					Feature normalized algorithms		
	FedAvg	FedBABU	SCAFFOLD	FedNTD	FedExP	FedETF	SphereFed	**FedDr+** (Ours)
Elapsed time	21.3	20.9	22.3	22.9	20.3	22.2	22.3	**24.4**

We compare **FedDr+** with various GFL algorithms for the elapsed time per communication round on CIFAR-100 (s=10). The results, detailed in Table 9, show that **FedDr+** exhibits a similar but slightly longer elapsed time than the other algorithms.