- Appendix -

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

The notations and pseudo code of **FedDr+** and **FedDr+ FT** are organized in Appendix A. In Appendix B, we provide a detailed explanation of the pulling and pushing gradients of the CE loss. In Appendix C, we provide a theoretical analysis of dot-regression, focusing on the feature vector gradient of the loss and its implications under the NTK framework, particularly for unobserved classes. The experimental setup is described in Appendix D, which includes code implementation, dataset descriptions, model specifications, optimizer settings, NIID partition, and the hyperparameter search process. Additional experimental results, including further analysis on the synergy effect and PFL, as well as results on IID dataset performance, scalability experiments, and stochastic client data settings, are presented in Appendix E.

A Notations, Pseudo Code of FedDr+ and FedDr+ FT

In this section, we first introduce the key notations used in our method and then present the pseudocode for **FedDr+** and **FedDr+ FT**. The pseudocode provides a clear and concise implementation guide for both global federated learning (GFL) with **FedDr+** and personalized federated learning (PFL) with **FedDr+ FT**.

A.1 Main Notations

To maintain clarity, Table 6 defines key indices, datasets, model parameters, and computations in alg and **FedDr+ FT**, forming the basis for our method and analysis.

Table 6: 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 $z(x; \boldsymbol{\theta})$

A.2 Pseudo Code of FedDr+ and FedDr+ FT

We now present the pseudocode for **FedDr+** and **FedDr+ FT**, outlining their key operations for global and personalized federated learning. The algorithm consists of two main stages:

Algorithm 1 **FedDr+**, **FedDr+ FT**

Input: Total rounds R, local epochs E, training dataset D^i_{train} for client i, sampled client set $N^{(r)} \subset [N]$ at round r, learning rate $\eta^{(r)}$ at round r

Initial Parameters: ETF Classifier $\mathbf{V}$, Initial global model parameters $\mathbf{\Theta}^g_0 = (\boldsymbol{\theta}^g_0, \mathbf{V})$

for $i = 1, \ldots, N$ **do**
  Server broadcasts $\mathbf{V}$ to client i

/ STEP 1: Get a GFL Model Θ^g_R of FedDr+ **/**

for $r = 1, \ldots, R$ **do**
  Server samples clients $N^{(r)}$ and broadcasts $\boldsymbol{\theta}^i_r \leftarrow \boldsymbol{\theta}^g_{r-1}$ **for** *each client* $i \in N^{(r)}$ ***in parallel*** **do**
    for *Local Steps* $e = 1, \ldots, E$ **do**
      for *Batches* $j = 1, \ldots, B$ **do**
        $\boldsymbol{\theta}^i_r \leftarrow \boldsymbol{\theta}^i_r - \eta^{(r)} \nabla \mathcal{L}_{\text{Dr+}}([D^i_{\text{train}}]_j; \boldsymbol{\theta}^i_r, \boldsymbol{\theta}^g_{r-1}, \boldsymbol{V})$ *Using [Equation (1)]*
    Upload $\boldsymbol{\theta}^i_r$ to server
  Server Aggregation: $\boldsymbol{\theta}^g_r \leftarrow \frac{1}{|N^{(r)}|} \sum_{i \in N^{(r)}} \boldsymbol{\theta}^i_r$

GFL output: $\mathbf{\Theta}^g_R = (\boldsymbol{\theta}^g_r, \boldsymbol{V})$

/ STEP 2: Get a PFL Models $\{\Theta^i_{R+1}\}^N_{i=1}$ of FedDr+FT **/**

for $i = 1, \ldots, N$ **do**
  Server broadcasts $\boldsymbol{\theta}^i_{R+1} \leftarrow \boldsymbol{\theta}^g_R$ to client i
  for *Local Steps* $e = 1, \ldots, E$ **do**
    for *Batches* $j = 1, \ldots, B$ **do**
      $\boldsymbol{\theta}^i_{R+1} \leftarrow \boldsymbol{\theta}^i_{R+1} - \eta^{(R)} \nabla \mathcal{L}_{\text{Dr+}}([D^i_{\text{train}}]_j; \boldsymbol{\theta}^i_R, \boldsymbol{\theta}^g_R, \boldsymbol{V})$ *Using [Equation (1)]*

PFL outputs: $\{\mathbf{\Theta}^i_{R+1} = (\boldsymbol{\theta}^i_{R+1}, \boldsymbol{V})\}^N_{i=1}$

B Preliminaries: Pulling and Pushing Feature Gradients in CE

In this section, we first compute the classifier's gradient with respect to the features. Next, we explain how the cross-entropy loss draws the pulling and pushing effects.

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

We begin by presenting two lemmas that support Proposition 1 and clarify pulling and pushing feature gradients in the cross-entropy (CE) loss.

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

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

(i) $c = c'$ case:

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

$$= p_c(x;\boldsymbol{\theta}) - p_c(x;\boldsymbol{\theta})^2 = p_c(x;\boldsymbol{\theta})(1 - p_c(x;\boldsymbol{\theta})).$$

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

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

□

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.

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

The last equality holds by 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.

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

Applying the chain rule for the second step and invoking Lemma 2 for the final equality confirms the result.

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

The gradient $\nabla_{f(x;\boldsymbol{\theta})}\mathcal{L}_{\mathrm{CE}}(x,y;\boldsymbol{\theta})$ consists of two components:

$$\mathbf{F}_{\mathrm{Pull}} = \big(1 - p_y(x;\boldsymbol{\theta})\big)v_y,$$
$$\mathbf{F}_{\mathrm{Push}} = -\sum_{c\in[C]\setminus\{y\}} p_c(x;\boldsymbol{\theta})v_c.$$

$\mathbf{F}_{\mathrm{Pull}}$ moves the feature vector towards the classifier vector v_y of the true class, promoting alignment. In contrast, $\mathbf{F}_{\mathrm{Push}}$ moves it away from the classifier vectors v_c for $c \in [C] \setminus \{y\}$, inducing misalignment.

□

C Theoretical Perspective of Dot-Regression (DR)

In this section, we provide a theoretical analysis of dot-regression (DR) loss in the context of feature-classifier alignment. We first derive the feature gradient of $\mathcal{L}_{\text{DR}}$ and analyze its effect on feature updates. We then present an NTK-based perspective explaining why dot-regression struggles with unobserved classes in FL. Finally, we compare DR with cross-entropy (CE) loss to highlight its limitations and the necessity of feature distillation.

C.1 Feature Gradient of $\mathcal{L}_{\text{DR}}$

In this subsection, we derive the gradient of dot-regression loss with respect to the feature vector on the observed classes.

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

$$\nabla_{f(x;\boldsymbol{\theta}_f)} \mathcal{L}_{\text{DR}}(x, y; \theta) = -\frac{1 - \cos\alpha}{\|f(x;\boldsymbol{\theta}_f)\|_2} \left\{ V_y - \cos\alpha \, \frac{f(x;\boldsymbol{\theta}_f)}{\|f(x;\boldsymbol{\theta}_f)\|_2} \right\},$$

where $\cos\alpha = \dfrac{f(x;\boldsymbol{\theta}_f)^\top}{\|f(x;\boldsymbol{\theta}_f)\|_2} V_y$.

Proof.

$$\begin{aligned}
\nabla_{f(x;\boldsymbol{\theta}_f)} \mathcal{L}_{\text{DR}}(x, y; \boldsymbol{\theta}) &= \nabla_{f(x;\boldsymbol{\theta}_f)} \left\{ \frac{1}{2} \left(\frac{f(x;\boldsymbol{\theta}_f)^T}{\|f(x;\boldsymbol{\theta}_f)\|_2} V_y - 1 \right)^2 \right\} \\
&= \left(\frac{f(x;\boldsymbol{\theta}_f)^T}{\|f(x;\boldsymbol{\theta}_f)\|_2} V_y - 1 \right) \nabla_{f(x;\boldsymbol{\theta}_f)} \frac{f(x;\boldsymbol{\theta}_f)^T}{\|f(x;\boldsymbol{\theta}_f)\|_2} V_y \\
&= \left(\frac{f(x;\boldsymbol{\theta}_f)^T}{\|f(x;\boldsymbol{\theta}_f)\|_2} V_y - 1 \right) \left[\frac{1}{\|f(x;\boldsymbol{\theta}_f)\|_2} \left\{ I - \frac{f(x;\boldsymbol{\theta}_f) f(x;\boldsymbol{\theta}_f)^T}{\|f(x;\boldsymbol{\theta}_f)\|_2^2} \right\} V_y \right] \\
&= -\frac{1 - \cos\alpha}{\|f(x;\boldsymbol{\theta}_f)\|_2} \left\{ V_y - \cos\alpha \frac{f(x;\boldsymbol{\theta}_f)}{\|f(x;\boldsymbol{\theta}_f)\|_2} \right\}.
\end{aligned}$$

□

C.1.1 Physical Meaning of $\nabla_{f(x;\theta)} \mathcal{L}_{\text{DR}}(x, y; \theta)$

According to Theorem C.1, the change in the feature vector $\Delta f(x; \boldsymbol{\theta}_f)$ is given by:

$$\Delta f(x; \theta_f) = \eta \, \frac{1 - \cos\alpha}{\|f(x;\theta_f)\|_2} \left(V_y - \cos\alpha \frac{f(x;\theta_f)}{\|f(x;\theta_f)\|_2} \right),$$

where η is the learning rate and α is the angle between the feature vector $f(x; \boldsymbol{\theta}_f)$ and the target vector V_y.

The term inside the parentheses, $V_y - \cos\alpha \frac{f(x;\boldsymbol{\theta}_f)}{\|f(x;\boldsymbol{\theta}_f)\|_2}$, represents a component orthogonal to $f(x; \boldsymbol{\theta}_f)$ that points towards V_y. This component adjusts $f(x; \boldsymbol{\theta}_f)$ to increase its cosine similarity with V_y while also expanding its norm.

The scaling factor $\frac{1-\cos\alpha}{\|f(x;\boldsymbol{\theta}_f)\|_2}$ determines the update magnitude. As training progresses, $f(x; \boldsymbol{\theta}_f)$ aligns more closely with V_y, reducing $1 - \cos\alpha$ and increasing $\|f(x; \boldsymbol{\theta}_f)\|_2$. Consequently, $\Delta f(x; \boldsymbol{\theta}_f)$ diminishes over time, reflecting convergence as the cosine similarity with V_y approaches its maximum.

Figure 8 illustrates this process, showing how the orthogonal component drives both the rotation and scaling of $f(x; \boldsymbol{\theta}_f)$ toward alignment with V_y.

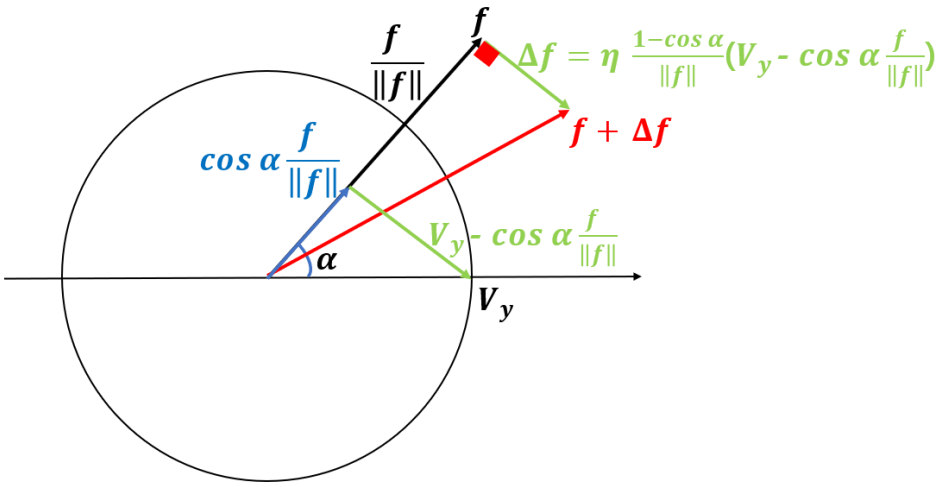


Figure 8: Feature gradient of $\mathcal{L}_{\mathrm{DR}}$. The gradient update rotates $f(x;\boldsymbol{\theta}_f)$ toward V_y while increasing its norm. As training progresses, the update magnitude decreases, leading to convergence.

C.2 NTK Perspective: Why Dot-Regression in FL Fails on Unobserved Classes

In this subsection, we analyze why dot-regression struggles with unobserved classes under the Neural Tangent Kernel (NTK) regime (Jacot et al., 2018). In the NTK regime, the feature gradient of any input is a weighted sum of the feature gradients from training samples. Assuming the network width is sufficiently wide, these weights depend only on the pair of inputs, the initialization distribution, such as He initialization (Glorot & Bengio, 2010b), and the activation functions.[§] The NTK regime holds when the setting where every layer in the neural network has infinite width, with parameters initialized i.i.d. This section explains how NTK-based gradient updates fail to align feature vectors with unobserved class directions, which leads to poor generalization in FL.

C.2.1 Gradient Flow in the NTK Regime

We treat gradient descent as a continuous process. P is the number of trainable parameters in the feature extractor, and θ_p ($p \in [P]$) denote each parameter. We focus on a specific client, denoted by i.

During training, gradient descent updates the model parameters to minimize the loss function. As below, we can see the evolution of the function $f(x;\boldsymbol{\theta}(t))$ can be analyzed using the kernel $\Theta^{(L)}(t)(x, x_i)$, which evolves along the training process:

$$
\begin{aligned}
\frac{\mathrm{d}f(x;\boldsymbol{\theta}(t))}{\mathrm{d}t} &= \sum_{p=1}^{P} \Big(\frac{\partial f(x;\boldsymbol{\theta}(t))}{\partial \theta_p}\Big)^{\top} \frac{\mathrm{d}\theta_p}{\mathrm{d}t} && \text{(Chain Rule)} \\
&= -\sum_{p=1}^{P} \Big(\frac{\partial f(x;\boldsymbol{\theta}(t))}{\partial \theta_p}\Big)^{\top} \frac{1}{|D^i_{\mathrm{train}}|} \sum_{(\tilde{x},\tilde{y})\in D^i_{\mathrm{train}}} \frac{\partial f(\tilde{x};\boldsymbol{\theta}(t))}{\partial \theta_p} \nabla_{f(x_i;\theta)} \mathcal{L}(\tilde{x},\tilde{y};\boldsymbol{\theta}) && \text{(Gradient Descent)} \\
&= -\frac{1}{|D^i_{\mathrm{train}}|} \sum_{(\tilde{x},\tilde{y})\in D^i_{\mathrm{train}}} \underbrace{\Big(\sum_{p=1}^{P} \Big(\frac{\partial f(x;\boldsymbol{\theta}(t))}{\partial \theta_p}\Big)^{\top} \frac{\partial f(\tilde{x};\boldsymbol{\theta}(t))}{\partial \theta_p}\Big)}_{\boldsymbol{\Theta}^{(L)}(t)(x,\tilde{x})\in\mathbb{R}^{d\times d}} \nabla_{f(\tilde{x};\theta)} \mathcal{L}(\tilde{x},\tilde{y};\boldsymbol{\theta}) \\
&= -\frac{1}{|D^i_{\mathrm{train}}|} \sum_{(\tilde{x},\tilde{y})\in D^i_{\mathrm{train}}} \boldsymbol{\Theta}^{(L)}(t)(x,\tilde{x}) \nabla_{f(\tilde{x};\theta)} \mathcal{L}(\tilde{x},\tilde{y};\boldsymbol{\theta})\,.
\end{aligned}
$$

[§]In practice, finite-width effects cause deviations from the ideal NTK behavior.

In the NTK regime with infinitely large widths, the matrix $\mathbf{\Theta}^{(L)}(t)(x, \tilde{x})$ converges to a scalar multiple of the identity matrix, $\Theta_\infty^{(L)}(x, \tilde{x})\mathbf{I}$. Furthermore, with the same condition, this scalar kernel remains constant throughout training (Jacot et al., 2018; Yang, 2020; Belfer et al., 2021), though finite-width effects may introduce small variations. In the NTK regime, the gradient descent dynamics are given by:

$$\frac{\mathrm{d}f(x;\boldsymbol{\theta}(t))}{\mathrm{d}t} = -\frac{1}{|D^i_{\text{train}}|}\sum_{(\tilde{x},\tilde{y})\in D^i_{\text{train}}} \underbrace{\Theta_\infty^{(L)}(x,\tilde{x})}_{\in\mathbb{R}} \nabla_{f(x_i;\theta)}\mathcal{L}(\tilde{x},\tilde{y};\boldsymbol{\theta})\,. \qquad \text{(in NTK Regime)}$$

Thus, $\Theta_\infty^{(L)}(x, \tilde{x})$ determines how each training sample $\tilde{x}$ influences an arbitrary input x, and in NTK regime, this weight depends only on the initialization distribution.

In Federated Learning (FL), local models are independently updated on different clients before aggregation. Under the NTK regime, each client follows the gradient flow during local training. FL aggregation then combines feature representations learned from different data distributions, leading to shifts in the global feature representation. By aggregating updates from multiple clients, FL integrates feature information from clients that have observed missing classes, thereby improving feature alignment.

C.2.2 Limitations of Dot-Regression loss in FL under the NTK Regime

Dot-regression loss (Yang et al., 2022) speeds up the alignment of feature vectors $f(x;\boldsymbol{\theta}) \in \mathbb{R}^d$ (pre-classifier layer outputs) with the true class direction v_y by minimizing the cosine angle:

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

This loss function is motivated by the decomposition of cross-entropy (CE) loss gradients into *pulling* and *pushing* components. Prior work suggests that removing the pushing effect in CE can improve convergence (Yang et al., 2022; Li & Zhan, 2021).

Let c be an unobserved class for a specific client i with the classifier vector v_c. From Theorem C.1, it follows that under the NTK regime, the gradient descent process on the client i is independent of v_c for arbitrary input x.

To analyze this, we first express the feature gradient under the dot-regression loss $\mathcal{L}_{\text{DR}}$ in the local learning stage. For simplicity, we omit the dependence on $\boldsymbol{\theta}(t)$ in the feature notation and write $\cos(f(\tilde{x};\boldsymbol{\theta}(t)), v_y)$ as $\cos(f(\tilde{x}), v_y)$. The feature gradient is given by:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}t} = \frac{1}{|D^i_{\text{train}}|}\sum_{y\in\mathcal{O}^i}\sum_{(\tilde{x},y)\in D^i_{\text{train}}} \Theta_\infty^{(L)}(x,\tilde{x})\frac{1-\cos(f(\tilde{x}),v_y)}{\|f(\tilde{x})\|_2}\Big(v_y - \cos(f(\tilde{x}),v_y)\frac{f(\tilde{x})}{\|f(\tilde{x})\|_2}\Big). \qquad \text{(in NTK Regime)}$$

Since $c \notin \mathcal{O}^i$, the feature gradient evaluated on training data does not depend on v_c. Given that feature gradients are a weighted sum over training data in the NTK regime, this implies that the learned feature representation for an arbitrary input remains unaffected by v_c during local training.

Therefore, dot-regression cannot align features with unobserved classes in local training. To examine this effect more closely, consider two cases $f_1(x)$ and $f_2(x)$ with the same input x with label c, whose settings and initialization at time $t = 0$ are identical except for the classifier vector v_c of class c, fixed with w and $-w$ ($\|w\| = 1$, $\forall y \in \mathcal{O}^i : w \perp v_y$). In the NTK regime under the dot-regression loss, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle f(x), v_c\rangle = -\frac{1}{|D^i_{\text{train}}|}\sum_{y\in\mathcal{O}^i}\sum_{(\tilde{x},y)\in D^i_{\text{train}}} \Theta_\infty^{(L)}(x,\tilde{x})\frac{\cos(f(\tilde{x}),v_y)(1-\cos(f(\tilde{x}),v_y))}{\|f(\tilde{x})\|_2^2}\langle f(\tilde{x}), v_c\rangle.$$

(in NTK Regime)

Since every term in the update equation is identical for $f_1(x)$ and $f_2(x)$, except for $\langle f(\tilde{x}), v_c\rangle$, which takes opposite values in each case, it follows that $\langle f_1(x), v_c\rangle = -\langle f_2(x), v_c\rangle$ for all time $t \geq 0$. This demonstrates

that classifier initialization strongly determines alignment in the local learning stage. Consequently, the global aggregation stage is the only way to generalize to classes that haven't been observed yet. This slows down the overall accuracy of the FL server.

C.2.3 Cross-Entropy Loss and Feature-Classifier Alignment

In contrast, cross-entropy (CE) loss explicitly guides feature gradients toward v_c, weighted by the softmax probability p_c and the NTK weight. This ensures that even when class c is absent, local training still produces meaningful updates. After each global aggregation, the refined p_c further strengthens alignment, allowing CE to maintain consistent feature-classifier alignment across all classes.

This observation aligns with our empirical findings: without feature distillation, dot-regression struggles to generalize to unobserved classes, whereas CE enables continuous feature updates, leading to improved generalization.

D Experimental Setup

This section details the code implementation, dataset descriptions, model specifications, optimizer settings, non-IID (NIID) partitioning, and hyperparameter search process used in our experiments.

D.1 Code Implementation

Our implementations are conducted using the PyTorch framework. Specifically, the experiments presented in Table 3 and Table 4 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`.

D.2 Datasets, Model, and Optimizer

To simulate a realistic FL scenario, we conduct extensive studies on three widely used datasets: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-100 Deng et al. (2009). For each dataset, appropriate models are employed: VGG11 (Simonyan & Zisserman, 2014) for CIFAR-10, MobileNet (Howard et al., 2017) for CIFAR-100, and ResNet-18 (He et al., 2016) for ImageNet-100. A momentum optimizer is utilized for all experiments. The data preprocessing pipeline for the training phase includes `RandomResizedCrop`, `RandomHorizontalFlip`, and `Normalize` transformations for all datasets. During testing, only the `Normalize` transformation is applied for CIFAR-10 and CIFAR-100, while for ImageNet-100, `Resize`, `CenterCrop`, and `Normalize` are applied. Unless otherwise noted, the basic setting of our experiments follows the dataset statistics, FL scenario specifications, and optimizer hyperparameters summarized in Table 7.

Table 7: Summary of Dataset, Model, FL System, and Optimizer Specifications

Datasets	C	$\lvert D_{\text{train}}\rvert$	$\lvert D_{\text{test}}\rvert$	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
ImageNet-100	100	130000	5000	100	320	0.1	5	50	0.9	1e-5

Note: In terms of dataset information, C represents the number of classes in the dataset, with $|D_{\text{train}}|$ and $|D_{\text{test}}|$ 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 the number of local epochs E were determined via extensive grid search for each algorithm, with details outlined in Appendix D.4.

D.3 Non-IID Partition Strategies

To induce heterogeneity in each client's training and test data $(D^i_{\text{train}}, D^i_{\text{test}})$, 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** (McMahan et al., 2017; Oh et al., 2022): We organize the D_{train} and D_{test} by label and divide them into non-overlapping shards of equal size. Each shard encompasses $\frac{|D_{\text{train}}|}{100\times s}$ and $\frac{|D_{\text{test}}|}{100\times s}$ samples of the same class, where s denotes the number of shards per client. This sharding technique is 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)** (Luo et al., 2021; Wang et al., 2020a): 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.

D.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, CIFAR-100, and ImageNet-100 datasets. The process and reasoning are outlined below.

D.4.1 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.

D.4.2 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.

D.4.3 Grid Search Process and Results

Considering the above reasons, we perform grid search for η and E on CIFAR-10, CIFAR-100, and ImageNet-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. Additionally, for ImageNet-100, a shard size of 20 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}. A default initial learning rate of 0.01 is used unless specified otherwise. The optimal learning rates vary by algorithm, and the results are summarized in Table 8. Table 8 also includes the additional hyperparameters used for each algorithm. The notation for these additional hyperparameters follows the conventions used throughout this paper (Li et al., 2020; Lee et al., 2022; Jhunjhunwala et al., 2023; Li et al., 2023b; Lee et al., 2024). 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}. For E, we consider {1, 3, 5, 10}. 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 9. Table 9 also includes the additional hyperparameters used for each algorithm. The notation for these additional hyperparameters follows the conventions used throughout this paper (Li et al., 2020; Lee et al., 2022; Jhunjhunwala et al., 2023; Li et al., 2023b; Lee et al., 2024). The optimal number of local epochs is found to be 3 for every algorithm.

ImageNet-100. We examine η values from $\{0.01, 0.1, 1.0, 10.0\}$, which are chosen to maintain a consistent logarithmic scale difference. 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 10. Table 10 also includes the additional hyperparameters used for each algorithm. The notation for these additional hyperparameters follows the conventions used throughout this paper (Li et al., 2020; Lee et al., 2022; Jhunjhunwala et al., 2023; Li et al., 2023b; Lee et al., 2024). The optimal number of local epochs is fixed at 5, following the setting of (Lee et al., 2024).

Table 8: Hyperparameters for VGG11 training on CIFAR-10.

	Feature un-normalized algorithms									Feature normalized algorithms		
Hyperparameters	FedAvg	FedBABU	FedProx	SCAFFOLD	MOON	FedNTD	FedExP	FedSOL	FedGELA	FedETF	SphereFed	**FedDr+** (Ours)
η	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.55	0.35
Additional	None	None	μ=0.001	None	(μ, τ)=(1,0.5)	(β, τ)=(1,3)	ϵ=0.001	ρ=2.0	None	(β, τ)=(1,1)	None	β=0.9

Table 9: Hyperparameters for MobileNet training on CIFAR-100.

	Feature un-normalized algorithms									Feature normalized algorithms		
Hyperparameters	FedAvg	FedBABU	FedProx	SCAFFOLD	MOON	FedNTD	FedExP	FedSOL	FedGELA	FedETF	SphereFed	**FedDr+** (Ours)
η	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	6.5	5.0
Additional	None	None	μ=0.001	None	(μ, τ)=(1,0.5)	(β, τ)=(1,3)	ϵ=0.001	ρ=2.0	None	(β, τ)=(1,1)	None	β=0.9

Table 10: Hyperparameters for ResNet-18 training on ImageNet-100.

	Feature un-normalized algorithms									Feature normalized algorithms		
Hyperparameters	FedAvg	FedBABU	FedProx	SCAFFOLD	MOON	FedNTD	FedExP	FedSOL	FedGELA	FedETF	SphereFed	**FedDr+** (Ours)
η	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	1.0	1.0	1.0
Additional	None	None	μ=0.001	None	(μ, τ)=(1,0.5)	(β, τ)=(1,3)	ϵ=0.001	ρ=2.0	None	(β, τ)=(1,1)	None	β=0.9

E Additional Experiment Results

This section provides additional experimental results, including analysis on the synergy effect, personalized FL (PFL), IID dataset performance, stochastic client data settings, and scalability experiments.

E.1 Synergy Effect Details

Table 11: 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	+MOON	+KD	+NTD	+LD	+FD	Baseline	+Prox	+MOON	+KD	+NTD	+LD	+FD
FedAvg	37.22	36.87	37.43	36.25	37.71	37.17	37.82	42.52	43.22	44.79	44.21	43.39	43.43	43.76
FedBABU	46.20	46.03	46.49	46.37	47.22	46.71	46.95	47.37	46.62	46.27	47.60	46.48	45.78	46.49
SphereFed	43.90	41.96	43.13	44.94	43.47	43.95	45.21	46.98	43.77	46.81	47.76	47.25	47.01	49.74
FedETF	32.42	31.87	34.30	32.76	32.65	32.25	32.77	46.27	45.71	45.98	46.67	46.16	45.91	46.47
FedGELA	29.17	28.69	28.80	29.11	28.84	29.36	30.33	27.11	29.03	28.09	28.45	29.62	29.41	29.75
Dot-Regression	42.52	41.95	44.72	47.45	48.32	47.52	**48.69**	42.72	46.35	50.36	49.47	50.36	49.28	**50.86**

Table 12: Optimal β value selected through grid search to achieve the best synergy of various FL algorithms and regularizers.

	Sharding ($s = 10$)							LDA ($\alpha = 0.1$)						
Algorithm	Baseline	+Prox	+MOON	+KD	+NTD	+LD	+FD	Baseline	+Prox	+MOON	+KD	+NTD	+LD	+FD
FedAvg	None	0.999	0.5	0.9999	0.9999	0.999	0.9	None	0.999	0.99	0.999	0.99	0.999	0.9999
FedBABU	None	0.9999	0.9	0.999	0.99	0.999	0.999	None	0.999	0.999	0.999	0.999	0.99	0.9999
SphereFed	None	0.9999	0.9999	0.9999	0.9	0.99	0.9	None	0.9999	0.999	0.999	0.9999	0.999	0.99
FedETF	None	0.999	0.3	0.5	0.999	0.9	0.9	None	0.9999	0.9999	0.5	0.999	0.99	0.99
FedGELA	None	0.9999	0.7	0.5	0.7	0.5	0.7	None	0.99	0.9	0.5	0.5	0.5	0.3
Dot-Regression	None	0.9999	0.9	0.5	0.5	0.5	0.9	None	0.9999	0.5	0.5	0.5	0.5	0.9

Table 13: Synergy of various FL algorithms and regularizers at $\beta = 0.9$.

	Sharding ($s = 10$)							LDA ($\alpha = 0.1$)						
Algorithm	Baseline	+Prox	+MOON	+KD	+NTD	+LD	+FD	Baseline	+Prox	+MOON	+KD	+NTD	+LD	+FD
FedAvg	37.22	30.27	36.67	35.14	35.56	34.83	37.82	42.52	36.09	42.09	41.48	41.34	43.36	43.10
FedBABU	46.20	36.71	46.49	45.50	45.09	45.81	45.31	47.37	39.04	45.92	45.58	45.56	46.46	44.77
SphereFed	43.90	1.36	1.89	41.01	43.47	41.73	45.21	46.98	1.46	2.21	45.22	46.25	43.84	48.61
FedETF	32.42	25.18	32.58	32.76	31.98	32.25	32.77	46.27	34.92	45.38	44.94	45.77	44.36	45.92
FedGELA	29.17	25.52	28.57	28.84	28.67	28.37	29.07	27.11	26.84	28.09	27.78	28.27	27.97	27.60
Dot-Regression	42.52	5.42	44.72	46.60	45.78	47.52	**48.69**	42.72	7.47	30.69	48.19	33.08	49.09	**50.79**

We evaluate the synergy effect of various FL algorithms by maintaining their original training loss while incorporating specific regularizers, as detailed in Equation 1 of the main text. To manage the differing loss scales between the baseline FL algorithms and the regularizers, we systematically tune the coefficient β across a range of values $(0.1, 0.3, 0.5, 0.7, 0.9, 0.99, 0.999, 0.9999)$. The resulting performance and optimal β values are shown in Table 11 and Table 12. However, when we set $\beta = 0.9$ without addressing the issue of differing loss scales, the performance results, presented in Table 13, reveal that several synergies are significantly inferior due to this oversight.

Table 14: 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
Dot-Regression	42.52	49.02	52.86	$30.31_{\pm 7.95}$	$37.52_{\pm 5.60}$	$47.08_{\pm 3.69}$
Dot-Regression FT ($\mathcal{L}_{\mathrm{DR}}$)	$80.84_{\pm(5.99)}$	$74.18_{\pm(5.78)}$	$56.84_{\pm(5.04)}$	$72.02_{\pm(6.80)}$	$66.96_{\pm(5.36)}$	$60.34_{\pm(3.66)}$
Dot-Regression FT ($\mathcal{L}_{\mathrm{Dr+}}$)	$80.82_{\pm(6.12)}$	$73.73_{\pm(5.75)}$	$56.69_{\pm(4.95)}$	$71.85_{\pm(7.03)}$	$66.59_{\pm(5.32)}$	$59.87_{\pm(3.65)}$
FedDr+ (ours)	48.69	51.00	53.23	$39.63_{\pm 9.12}$	$45.83_{\pm 6.18}$	$48.04_{\pm 3.44}$
FedDr+ FT ($\mathcal{L}_{\mathrm{DR}}$) (ours)	$\mathbf{84.23}_{\pm(5.44)}$	$\mathbf{75.73}_{\pm(4.79)}$	$\mathbf{56.90}_{\pm(4.85)}$	$\mathbf{78.65}_{\pm(6.17)}$	$\mathbf{74.86}_{\pm(4.77)}$	$\mathbf{62.47}_{\pm(3.72)}$
FedDr+ FT ($\mathcal{L}_{\mathrm{Dr+}}$) (ours)	$84.10_{\pm(5.43)}$	$75.42_{\pm(4.80)}$	$56.76_{\pm(4.91)}$	$78.55_{\pm(6.16)}$	$74.75_{\pm(4.75)}$	$62.16_{\pm(3.73)}$

E.2 Personalized Federated Learning Results

We introduce **FedDr+** FT and dot-regression FT, inspired by prior work (Oh et al., 2022; Dong et al., 2022; Li et al., 2023b; Kim et al., 2023). These methods enhance personalization by leveraging local data to fine-tune the GFL model. We investigate the impact of fine-tuning using $\mathcal{L}_{\mathrm{Dr+}}$ and $\mathcal{L}_{\mathrm{DR}}$ 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 14 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 $\mathcal{L}_{\mathrm{DR}}$ or $\mathcal{L}_{\mathrm{Dr+}}$. This results in four combinations: Dot-Regression FT ($\mathcal{L}_{\mathrm{DR}}$), Dot-Regression FT ($\mathcal{L}_{\mathrm{Dr+}}$), **FedDr+** FT ($\mathcal{L}_{\mathrm{DR}}$), and **FedDr+** FT ($\mathcal{L}_{\mathrm{Dr+}}$). When the GFL model is fixed, using $\mathcal{L}_{\mathrm{DR}}$ for fine-tuning consistently outperforms $\mathcal{L}_{\mathrm{Dr+}}$ across all settings, because dot-regression focuses on local alignment which advantages personalized fine-tuning. Conversely, when the fine-tuning method is fixed, employing $\mathcal{L}_{\mathrm{Dr+}}$ for the GFL model consistently outperforms $\mathcal{L}_{\mathrm{DR}}$ across all settings. This aligns with previous research (Nguyen et al., 2022; Chen et al., 2023) suggesting that fine-tuning from a well-initialized model yields better PFL performance.

E.3 IID Data Performance

To address the question regarding the performance of **FedDr+** or dot-regression loss in Federated Learning (FL) settings with IID data, we conducted experiments on CIFAR-100 with 100 clients, distributing data IID and ensuring a fair number of samples per client. We evaluated FedAvg, FedBABU, Dot-regression, and **FedDr+** across 5 seeds, calculating the mean and standard deviation of the global model accuracy for each algorithm.

Table 15: Global model accuracy (%) in IID data settings.

Algorithm	**Accuracy (mean ± std)**
FedAvg	47.19 ± 1.06
FedBABU	45.18 ± 0.61
Dot-regression	51.48 ± 0.99
FedDr+	**51.10 ± 0.61**

From the Table 15, it is evident that Dot-regression and **FedDr+** achieve the highest performance, significantly outperforming both FedAvg and FedBABU. The performance of Dot-regression and **FedDr+** is nearly identical under IID settings.

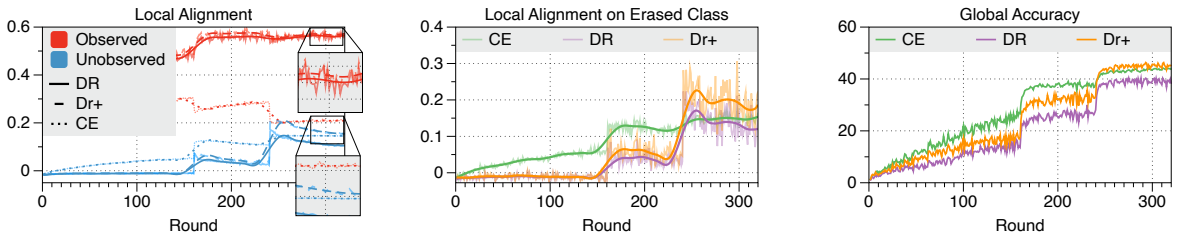
This similarity arises because, in the IID scenario, there are no **unobserved classes** across clients. As a result, the feature distillation mechanism in **FedDr+**, which is specifically designed to mitigate forgetting on unobserved classes, does not provide additional benefits. Instead, both Dot-regression and **FedDr+** excel in improving local alignment across all classes, fully achieving the global model's objective of enhancing local alignment for all clients.

E.4 Performance in Stochastic Client Data Settings

While our original experiments on **CIFAR-100 (s=10) with 100 clients** assumed a static client dataset, we conducted additional experiments where each client randomly removed one class from its dataset every 10 FL rounds. As expected, global model accuracy decreased for all methods, as shown in Table 16. However, **FedDr+** consistently outperformed CE and Dot-regression, demonstrating its robustness in handling dynamic class distributions. The round-wise global test accuracy trends for CE, Dot-regression, and **FedDr+** in the stochastic setting are presented in Figure 9c, further confirming **FedDr+** 's stability and superior performance across training rounds.

Table 16: Global model accuracy (%) in static and stochastic client data settings.

Algorithm	Static Setting	Stochastic Setting
CE	46.20	43.59
Dot-regression	42.52	38.13
FedDr+	**48.69**	**44.96**



(a) Local alignment (b) Local alignment on erased class (c) Global test accuracy

Figure 9: Comparison of (a) feature-classifier alignment on the observed and unobserved classes test data, (b) feature-classifier alignment on erased-class test data for $\boldsymbol{\theta}_r^i$, and (c) global test accuracy of $\boldsymbol{\theta}_r^g$ on all classes. Models are trained using $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{DR}}$, and $\mathcal{L}_{\text{Dr+}}$.

To further investigate why **FedDr+** maintains superior global accuracy in the stochastic setting, we analyzed the feature-classifier alignment for both observed/unobserved classes and erased classes.

- **Local alignment for observed/unobserved classes (Fig 9a):**
  - **FedDr+** maintains superior feature-classifier alignment for both observed and unobserved classes compared to Dot-regression, consistently outperforming it across all rounds.
  - During the final convergence phase, **FedDr+** surpasses even CE in unobserved class alignment, confirming its effectiveness in preserving global knowledge.
- **Local alignment for erased class (Fig 9b):**
  - Even for erased class (those removed during training), **FedDr+** retains stronger feature-classifier alignment than Dot-regression.
  - During the final convergence phase, **FedDr+** also surpasses CE in erased class alignment, further demonstrating its ability to mitigate forgetting of removed class knowledge.

These results suggest that the **feature distillation mechanism in FedDr+ effectively enhances global knowledge preservation while also enabling effective learning of observed classes, even when class distributions change dynamically**.

E.5 Scaling to Larger Numbers of Clients and Training Rounds

We conducted experiments on **CIFAR-100 (s=10)** with **1,000 communication rounds**, increasing the number of clients to 100, 200, 500, and 1,000. All algorithms used previously grid-searched optimal hyperparameters, and results are averaged over three independent seeds. All algorithms used previously grid-searched optimal hyperparameters, and results are averaged over three independent seeds.

Table 17: Global model accuracy (%) for different numbers of clients with 1,000 communication rounds.

Algorithm	N=100	N=200	N=500	N=1,000
FedAvg	50.50 ± 0.57	42.51 ± 1.47	33.02 ± 0.74	26.63 ± 1.31
FedBABU	58.19 ± 1.07	48.75 ± 1.99	37.40 ± 0.41	25.10 ± 1.08
FedDr+	**64.21 ± 1.24**	**59.78 ± 0.71**	**43.27 ± 0.31**	**28.99 ± 0.98**

Table 17 confirms that **FedDr+** consistently outperforms FedAvg and FedBABU across all settings, demonstrating robust scalability in large-scale FL.