Guided Model Merging for Hybrid Data Learning: Leveraging Centralized Data to Refine Decentralized Models

Supplementary Material

A. Proof

In this section, we prove the convergence rate of `Feddle` when $\mathcal{D}_S$ is either the in-domain or out-of-domain data. For notational simplicity we let $\Delta_j^k = \Delta\boldsymbol{\omega}_j^k$ for the updates of the j-th client and $\boldsymbol{\Delta}^k = (\Delta_1^k, \dots, \Delta_M^k)$ for some $M > 0$. We let ∇F denotes the usual gradient on $\boldsymbol{\omega}$ and $\nabla_{\boldsymbol{c}} F(\boldsymbol{\omega} + \sum_{m=1}^M \hat{c}_m \Delta_m)$ for the gradient with respect to $\boldsymbol{c}$ where Δ_m is the model update of client m.

We first prove a useful lemma for the smoothness of the gradient $\nabla F_{\boldsymbol{c}}$.

Lemma 1. *Suppose Assumptions 1-4 hold, then $\tilde{F}(\boldsymbol{c}) := F(\boldsymbol{\omega} + \sum_{m=1}^M \boldsymbol{c}_m \Delta_m)$ has $L(\sum_{m=1}^M ||\Delta_m||^2)$-smoothness with respect to $\boldsymbol{c}$. Therefore, for the convergence of the training on the server, $\eta_{\boldsymbol{c}} \le \mathcal{O}\left(\frac{1}{L\sum_{m=1}^M ||\Delta_m||^2}\right)$.*

Proof. Let $\boldsymbol{\omega}(\boldsymbol{c}) = \boldsymbol{\omega} + \sum_{m=1}^M \boldsymbol{c}_m \Delta_m$. Then, we have by definition that

$$\begin{aligned}
&||\nabla \tilde{F}_{\boldsymbol{c}}(\boldsymbol{c}) - \nabla \tilde{F}_{\boldsymbol{c}}(\boldsymbol{c}')|| \\
\le& ||((\nabla F(\boldsymbol{\omega}(\boldsymbol{c})) - \nabla F(\boldsymbol{\omega}(\boldsymbol{c}')) \cdot \Delta_m)_m|| \\
\le& ||\nabla F(\boldsymbol{\omega}(\boldsymbol{c})) - \nabla F(\boldsymbol{\omega}(\boldsymbol{c}'))||(\sum_m ||\Delta_m||^2)^{1/2} \\
\le& L||(c - c') \cdot (\Delta_1, \dots, \Delta_M)||(\sum_m ||\Delta_m||^2)^{1/2} \\
\le& L(\sum_m ||\Delta_m|^2)||c - c'||.
\end{aligned}$$

□

Proof of Theorem 1. We first discuss the relation between our merging method $\sum_m \hat{c}_m \boldsymbol{a}_m$ and that of FedBuff or `FedAvg`. For each k, let $\tilde{\boldsymbol{\omega}}^{k+1}$ be the weight update from FedBuff or `FedAvg` based on $\boldsymbol{\omega}^k$. In FedBuff, $\tilde{\boldsymbol{\omega}}^{k+1} = \frac{\eta_g}{M} \sum_{j \in \mathcal{M}} \Delta_j^{k-\tau_k^j}$ where $\mathcal{M}$ is the recent sampled set of clients and τ_k^j is the delay. Since `Feddle` has the same sampling procedure and these gradients from new clients are added to the atlas, there must exist certain $\hat{c}_m$ such that $\sum_m \hat{c}_m \boldsymbol{a}_m = \tilde{\boldsymbol{\omega}}^{k+1}$. In other words, `Feddle`'s output in this step $\boldsymbol{\omega}^{k+1}$ must have a lower error than $\tilde{\boldsymbol{\omega}}^{k+1}$. In `FedAvg`, $\tilde{\boldsymbol{\omega}}^{k+1} = \frac{\eta_g |\mathcal{D}_j|}{|\mathcal{D}|} \sum_{j=1}^J \Delta_j^k$, where each client's gradient is involved. For this, we define $\hat{c}_m \propto \eta_g |\mathcal{D}_j| / |\mathcal{D}|$ (by $\propto$ we resume the unnormalized gradient) if $\boldsymbol{a}_m$ is the most recent gradient of each client and otherwise 0. By the L-smoothness assumption, the difference in the error at $\tilde{\boldsymbol{\omega}}^{k+1}$ and the defined $\boldsymbol{\omega}^k + \sum_m \hat{c}_m \boldsymbol{a}_m$ can be bounded by $\sum_j L||\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k-\tau_k^j}||$ (their gradients are calculated at $\boldsymbol{\omega}^k$,$\boldsymbol{\omega}^{k-\tau_k^j}$). According to the proof of Theorem 3.4 of [54], such difference can be bounded by Eq.(C.8), and their overall order is less than $\eta_g^2 \eta_l^2$, which is therefore absorbed in -$\eta_g \eta_l ||\nabla F(\boldsymbol{\omega}^k)||^2$ for sufficiently small parameters. Thus, we can safely assume that the same conclusion holds when comparing `Feddle` and `FedAvg` up to a negligible error.

Now, we consider the expected loss on $\mathcal{D}_S$ (and thus $\mathcal{D}$), and $\mathcal{D}_j$, $1 \le j \le J$, we have

$$\begin{aligned}
F(\boldsymbol{\omega}^{k+1}) - F(\boldsymbol{\omega}^k) = F(\boldsymbol{\omega}^{k+1}) - F(\tilde{\boldsymbol{\omega}}^{k+1}) \\
+ F(\tilde{\boldsymbol{\omega}}^{k+1}) - F(\boldsymbol{\omega}^k).
\end{aligned}$$

Based on the proceeding argument, $F(\boldsymbol{\omega}^{k+1}) \le F(\tilde{\boldsymbol{\omega}}^{k+1})$ holds, thus we can assume without loss of generality that the optimization on $\boldsymbol{c}$ is initialized at the $\boldsymbol{c}$ such that $\tilde{w}^{k+1} = \boldsymbol{\omega}^k + \sum_m \hat{c}_m \boldsymbol{a}_m$. By the existing convergence results of FL methods [54], we can upperbound $F(\tilde{\boldsymbol{\omega}}^{k+1}) - F(\boldsymbol{\omega}^k)$ by $-C_1 \mathbb{E}[||\nabla F(\boldsymbol{\omega}^k)||^2], C_1 > 0$ plus some constants regarding T, K. Thus, if we can upper bound $F(\boldsymbol{\omega}^{k+1}) - F(\tilde{\boldsymbol{\omega}}^{k+1})$ by $-C_2 \mathbb{E}[||\nabla F(\boldsymbol{\omega}^k)||^2], C_2 > 0$, then we can indeed show that the convergence rate of `Feddle` is faster at least by some factor. Now, we investigate the optimization of $\boldsymbol{c}$ in Eq.(5). Since it uses SGD and the L- smoothness holds, we can upperbound the loss reduction by the sum of the squared L^2 norm of the gradients at each $\boldsymbol{c}$. In particular, we have

$$\begin{aligned}
F(\boldsymbol{\omega}^{k+1}) - F(\tilde{\boldsymbol{\omega}}^{k+1}) \le F(\boldsymbol{\omega}_{\boldsymbol{c}_1}^{k+1}) - F(\tilde{\boldsymbol{\omega}}^{k+1}) \\
\le -\frac{\eta_{\boldsymbol{c}}}{2} ||\nabla_{\boldsymbol{c}} F(\tilde{\boldsymbol{\omega}}^{k+1})||^2 + \frac{\eta_{\boldsymbol{c}}^2 \sigma_g'^2}{2},
\end{aligned}$$

where $\boldsymbol{\omega}_{\boldsymbol{c}_1}^{k+1}$ is the weight update after the first step of the optimization initialized by $\tilde{\boldsymbol{\omega}}^{k+1}$, and $\sigma_g'^2$ is the noise of the gradient, which we can also assume to be controlled by the squared norm of the gradient. Next, we would like to relate the gradient regarding $\boldsymbol{c}$ with the gradient at $\boldsymbol{\omega}^k$ to estimate C_2 as previously discussed. We assumed that $\tilde{\boldsymbol{\omega}}^{k+1} = \frac{\eta_g}{J} \sum_{j \in J} \Delta_j^k$ for some $J > 0$, which is a unified expression for FL methods (if M anchors are actually used, simply let some $\Delta_j^k = 0$ and change J to M in the following). By the chain rule of differentiation, we have that

$$
\begin{aligned}
&||\nabla_{\boldsymbol{c}} F(\tilde{\boldsymbol{\omega}}^{k+1})||^2 \\
=&||(\nabla F(\tilde{\boldsymbol{\omega}}^{k+1})^\top \Delta_1, \dots, F(\tilde{\boldsymbol{\omega}}^{k+1})^\top \Delta_J)||^2 \\
=&\sum_{j=1}^{J} (\langle \nabla F(\boldsymbol{\omega}^k), \Delta_j \rangle + \langle \nabla F(\tilde{\boldsymbol{\omega}}^{k+1}) - \nabla F(\boldsymbol{\omega}^k), \Delta_j \rangle)^2 \\
\geq &\sum_{j=1}^{J} (\langle \nabla F(\boldsymbol{\omega}^k), \Delta_j \rangle)^2 \\
&- 2\eta_l L ||\nabla F(\boldsymbol{\omega}^k)|| ||\tilde{\boldsymbol{\omega}}^{k+1} - \boldsymbol{\omega}^k|| ||\boldsymbol{\Delta^k}||^2 \\
\geq &\frac{J}{\eta_g^2} (\langle \nabla F(\boldsymbol{\omega}^k), \frac{\eta_g}{J} \sum_{j=1}^{J} \Delta_j \rangle)^2 - \\
&2\eta_l L ||\nabla F(\boldsymbol{\omega}^k)|| ||\tilde{\boldsymbol{\omega}}^{k+1} - \boldsymbol{\omega}^k|| ||\boldsymbol{\Delta^k}||^2,
\end{aligned}
$$

where the last inequality used $J \sum_j a_j^2 \geq (\sum_j a_j)^2$. Since $\frac{\eta_g}{J} \sum_{j=1}^{J} \Delta_j^k$ is the update in the FedBuff (or `FedAvg` depending on the mechanism), we can apply their theory to calculate the lower bound of the first term on the right-hand side. In particular, by Eq. (C.2)-(C.6) of [54] and our condition on the gradient norm, we have that

$$
\begin{aligned}
&|\langle \nabla F(\boldsymbol{\omega}^k), \frac{\eta_g}{J} \sum_{j=1}^{J} \Delta_j \rangle| \geq \frac{\eta_g \eta_l Q}{2} ||\nabla F(\boldsymbol{\omega}^k)||^2 \\
&- \frac{A_1 \eta_g \eta_l Q^2 \eta_l^2 L^2 (C+1)}{J} (\sum_{q,j} (||\nabla F(\boldsymbol{\omega}^{k-\tau_k^j})||^2 \\
&+ 4M \hat{c}_{max}^2 \tau_{max}^2 \max_{k-\tau_k^j \leq s \leq k} ||\nabla F_j(\boldsymbol{\omega}^s)||^2)) \\
&\geq \frac{\eta_g \eta_l Q}{2} ||\nabla F(\boldsymbol{\omega}^k)||^2 \times \\
&\quad \left(1 - A_1 Q^2 \eta_l^2 L^2 (C+1) C_{max} (1 + 4\hat{c}_{max}^2 \tau_{max}^2)\right)
\end{aligned}
$$

for some constants A_1. Here, we used $\frac{1}{J} \sum_j ||\nabla F_j(\boldsymbol{\omega})||^2 \leq \frac{2}{J} \sum_j ||\nabla F_j(\boldsymbol{\omega}) - \nabla F(\boldsymbol{\omega})||^2 + 2||\nabla F(\boldsymbol{\omega})||^2$ in the last inequality. Note that $\eta_l Q$ is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$, and $\hat{c}_{max} \tau_{max} = \mathcal{O}(1)$, so the coefficients must be positive. Now, for the second term, we have by Eq.(C.6) of [54] and our condition on the signal-to-noise ratio that

$$
\begin{aligned}
&||\tilde{\boldsymbol{\omega}}^{k+1} - \boldsymbol{\omega}^k||^2 \\
\leq &\frac{2(C+1)Q\eta_g^2\eta_l^2}{M} \sum_{q=0}^{Q-1} ||\nabla F_j(\boldsymbol{\omega}^{k-\tau_k^j, q})||^2 \\
\leq &\frac{2(C+1)Q\eta_g^2\eta_l^2}{M} \sum_{q=0}^{Q-1} (1 + C\eta_l LG)^{2q} \sum_j ||\nabla F_j(\boldsymbol{\omega}^{k-\tau_k})||^2 \\
\leq &\frac{2(C+1)C_{max}Q\eta_g^2\eta_l^2}{M} \sum_{q=0}^{Q-1} (1 + C\eta_l LG)^{2q} \sum_j ||\nabla F_j(\boldsymbol{\omega}^k)||^2 \\
\leq &\frac{2(C+1)C_{max}Q\eta_g^2\eta_l^2}{M} \sum_{q=0}^{Q-1} (1 + C\eta_l LG)^{2q} \sum_j ||\nabla F_j(\boldsymbol{\omega}^k) \\
&- \nabla F(\boldsymbol{\omega}^k) + \nabla F(\boldsymbol{\omega}^k)||^2 \\
\leq &A_2 (C+1)^2 C_{max} Q \eta_g^2 \eta_l^2 ||\nabla F(\boldsymbol{\omega}^k)||^2.
\end{aligned}
$$

Here, A_2 is a constant. The second inequality is due to the fact that each $\Delta_{j,q}$ is obtained by local SGD, and therefore, the changes in the gradient can be bounded by the initial value and some constants. The last inequality is due to Assumption 2, the low signal-to-noise ratio of the gradient and the order of η_l. In total, we have by choosing suitable coefficients for η_l that

$$
\begin{aligned}
&||\nabla_{\boldsymbol{c}} F(\tilde{\boldsymbol{\omega}}^{k+1})||^2 \geq \frac{J\eta_l^2 Q^2}{4} ||\nabla F(\boldsymbol{\omega}^k)||^4 \\
&- 2\eta_l^2 \eta_g Q L (C+1) \sqrt{A_2 C_{max}} ||\nabla F(\boldsymbol{\omega}^k)||^2 ||\boldsymbol{\Delta^k}||^2 \\
&\geq \left(\frac{A_3}{C_{max}} - 2\eta_l^2 \eta_g Q L (C+1) \sqrt{A_2 C_{max}} \right) \\
&\quad \cdot ||\nabla F(\boldsymbol{\omega}^k)||^2 ||\boldsymbol{\Delta^k}||^2. \qquad (13)
\end{aligned}
$$

Here, A_3 is a constant. We used Eq. (C.5) of [54] and our conditions on the gradient norm for the last inequality, which yield that $C_{max} \eta_l^2 Q^2 J ||\nabla F(\boldsymbol{\omega}^k)||^2 \geq A_3 ||\boldsymbol{\Delta^k}||^2$. $\eta_l^2 \eta_g Q = \mathcal{O}\left(\frac{M}{K}\right)$ which is assured to vanish given the convergence rate of federated learning algorithms, so the coefficients would be positive.

Since $||\boldsymbol{\Delta^k}||^2 = \sum_j ||\Delta_j||^2$, the decrease in the true loss is expected to be a constant factor of a function of $||\nabla F(\boldsymbol{\omega}^k)||^2$ and server steps T. Note that by using SGD under L'-smoothness, the norm of the gradient is at least $(1 - L'(1+C)\eta_{\boldsymbol{c}})^T ||\nabla F_{\boldsymbol{c}}(\tilde{\boldsymbol{\omega}}^{k+1})||$, therefore, $F(\boldsymbol{\omega}^{k+1}) - F(\boldsymbol{\omega}^k)$ is at least be upper bounded by

$$
-\eta_{\boldsymbol{c}} \frac{1 - (1 - L'(1+C)\eta_{\boldsymbol{c}})^{2T}}{1 - (1 - L'(1+C)\eta_{\boldsymbol{c}})^2} ||\nabla F_{\boldsymbol{c}}(\tilde{\boldsymbol{\omega}}^{k+1})||^2. \qquad (14)
$$

By the choice of $\eta_{\boldsymbol{c}}$ in Lemma 1 and our lower bound on $||\nabla F_{\boldsymbol{c}}(\tilde{\boldsymbol{\omega}}^{k+1})||^2$ in Eq. (13), we can upper bound the above by $-A_0(1 - \eta_l^2 \eta_g Q)\left(1 - \frac{1}{4^T}\right) ||\nabla F_(\boldsymbol{\omega}^k)||^2$, where A_0 is a constant decreasing in C, C_{max}, L. Therefore, the effective constants in front of $||\nabla F(\boldsymbol{\omega}^k)||^2$ is at least of order

$-A_0(1-\eta_l^2\eta_g Q)\left(1-\frac{1}{4^T}\right)/2-\eta_g\eta_l Q/2$ where the second constant comes from FedBuff [54]. We obtain the results by plugging in the parameters.

□

Proof of Theorem 2. Denote the expected surrogate loss by F^h, and the weight update after the t-th server step on F^h and F (i.e. out-of-domain and in-domain data) by $\boldsymbol{\omega}_{h^t}^k, \boldsymbol{\omega}_{f^t}^k$. We first have

$$||F(\boldsymbol{\omega}_{h^t}^k) - F(\boldsymbol{\omega}_{f^t}^k)|| \leq G||\boldsymbol{\omega}_{h^t}^k - \boldsymbol{\omega}_{f^t}^k||.$$

Since we assume that `Feddle` is initialized by FedBuff when the server data is out-of-domain, we can let $\boldsymbol{\omega}_{h^0}^k = \boldsymbol{\omega}_{f^0}^k$ at the start point. Now, for $t=1$, by the definition of $\eta_{\boldsymbol{c}}'$ and the cosine similarity between the gradients, it holds that

$$\begin{aligned}&||\boldsymbol{\omega}_{h^1}^k - \boldsymbol{\omega}_{f^1}^k|| = ||\boldsymbol{\omega}_{h^1}^k - \boldsymbol{\omega}_{h^0}^k + \boldsymbol{\omega}_{f^0}^k - \boldsymbol{\omega}_{f^1}^k||\\ =&\eta_{\boldsymbol{c}}||\eta_{\boldsymbol{c}}'\nabla_{\boldsymbol{c}}F^h(\boldsymbol{\omega}_{h^0}^k)/\eta_{\boldsymbol{c}} - \nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{f^0}^k)||\\ \leq&\eta_{\boldsymbol{c}}(1-s)||\nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{f^0}^k)||\\ \leq&\eta_{\boldsymbol{c}}(1-s)||\nabla F(\boldsymbol{\omega}_{f^0}^k)||||\boldsymbol{\Delta}^{\boldsymbol{k}}|| \ll 1.\end{aligned}$$

Here, the last equality follows from the definition of the gradient with respect to the merging coefficients and Cauchy-Schwartz inequality. In general, we have

$$\begin{aligned}&||\boldsymbol{\omega}_{h^t}^k - \boldsymbol{\omega}_{f^t}^k||\\ =&||\eta_{\boldsymbol{c}}'\nabla_{\boldsymbol{c}}F^h(\boldsymbol{\omega}_{h^{t-1}}^k) - \eta_{\boldsymbol{c}}\nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{f^{t-1}}^k) - \boldsymbol{\omega}_{f^{t-1}}^k + \boldsymbol{\omega}_{h^{t-1}}^k||\\ \leq&\eta_{\boldsymbol{c}}||\eta_{\boldsymbol{c}}'\nabla_{\boldsymbol{c}}F^h(\boldsymbol{\omega}_{h^{t-1}}^k)/\eta_{\boldsymbol{c}} - \nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{h^{t-1}}^k)||\\ &+ ||\boldsymbol{\omega}_{f^{t-1}}^k - \boldsymbol{\omega}_{h^{t-1}}^k|| + \eta_{\boldsymbol{c}}||\nabla F(\boldsymbol{\omega}_{h^{t-1}}^k) - \nabla F(\boldsymbol{\omega}_{f^{t-1}}^k)||\\ \leq&\eta_{\boldsymbol{c}}(1-s)||\nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{h^{t-1}}^k)||\\ &+(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})||\boldsymbol{\omega}_{f^{t-1}}^k - \boldsymbol{\omega}_{h^{t-1}}^k||\\ \leq&\eta_{\boldsymbol{c}}(1-s)G||\boldsymbol{\Delta}^{\boldsymbol{k}}|| + (1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})||\boldsymbol{\omega}_{f^{t-1}}^k - \boldsymbol{\omega}_{h^{t-1}}^k||\end{aligned}$$

And therefore by iteratively applying the formula, we have,

$$\begin{aligned}&||\boldsymbol{\omega}_{h^t}^k - \boldsymbol{\omega}_{f^t}^k||\\ \leq&\eta_{\boldsymbol{c}}(1-s)[G||\boldsymbol{\Delta}^{\boldsymbol{k}}||\sum_{t=0}^{T-1}(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})^t\\ &+(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})^t||\nabla_{\boldsymbol{c}}F(\boldsymbol{\omega}_{f^0}^k)||]\\ \leq&\eta_{\boldsymbol{c}}(1-s)G||\boldsymbol{\Delta}^{\boldsymbol{k}}||\sum_{t=0}^{T}(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})^t\end{aligned}$$

Therefore, we have an upper bound for $F(\boldsymbol{\omega}_{h^t}^k)$ by

$$F(\boldsymbol{\omega}_{h^t}^k) \leq F(\boldsymbol{\omega}_{f^t}^k) + \eta_{\boldsymbol{c}}(1-s)G||\boldsymbol{\Delta}^{\boldsymbol{k}}||\sum_{t=0}^{T}(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})^t$$

which means that

$$\begin{aligned}&F(\boldsymbol{\omega}_h^{k+1}) - F(\boldsymbol{\omega}^k) \leq\\ &F(\boldsymbol{\omega}^{k+1}) - F(\boldsymbol{\omega}^k) + \eta_{\boldsymbol{c}}(1-s)G||\boldsymbol{\Delta}^{\boldsymbol{k}}||\sum_{t=0}^{T}(1+L||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2\eta_{\boldsymbol{c}})^t\end{aligned}$$

Now, by choosing $\eta_{\boldsymbol{c}}$ adaptively such that $\eta_{\boldsymbol{c}} = \min\left(\frac{1}{LT||\boldsymbol{\Delta}^{\boldsymbol{k}}||^2}, \frac{1}{LT||\boldsymbol{\Delta}^{\boldsymbol{k}}||}\right)$, we have, no matter the magnitude of $||\boldsymbol{\Delta}^{\boldsymbol{k}}||$, that

$$\begin{aligned}&F(\boldsymbol{\omega}_h^{k+1}) - F(\boldsymbol{\omega}^k)\\ \leq&F(\boldsymbol{\omega}^{k+1}) - F(\boldsymbol{\omega}^k) + (1-s)\frac{A_4G}{TL}T\\ =&F(\boldsymbol{\omega}^{k+1}) - F(\boldsymbol{\omega}^k) + A_4(1-s)\frac{G}{L}.\end{aligned}$$

with a potential price that the C_T in the rate of `Feddle` changes to

$$C_T' = \frac{A_0}{T||\boldsymbol{\Delta}^{\boldsymbol{k}}||} \geq \frac{A_5}{TQ\eta_l(\sigma_l+\sigma_g+G)} \geq \frac{A_0'\sqrt{K}}{T(\sigma_l+\sigma_g+G)},$$

due to the change of $\eta_{\boldsymbol{c}}$ in Eq. (14), which still guarantees faster convergence. Therefore, we obtain the result by aggregating the loss reduction. □

B. Algorithm

The algorithm of `Feddle` is presented in Alg. 1.

C. Related Work

We begin by discussing related work on learning from decentralized data (App. C.1), with a particular emphasis on Federated Learning (FL). Next, we examine related work on learning from hybrid data in App. C.2. Finally, we discuss model averaging methods from a broad scope in App. C.3.

C.1. Learning from Decentralized Data

Next, we proceed to discuss learning from decentralized data, starting with related work on FL. We then briefly introduce other decentralized learning schemes.

Federated Learning The pioneering FL framework was proposed by McMahan et al. [41], which features a client-server architecture. In this setup, the server orchestrates the training process by sending the latest global model to connected clients for local training and aggregating their model updates once local training is completed at each client. The clients are responsible for managing local training using their own data and computational resources.

- *Data Heterogeneity.* One major challenge in FL is data heterogeneity, where clients are often geographically distant or environmentally distinct from each other [25, 34].

Algorithm 1 `Feddle`

```
Input: Server learning rate $\eta_g$, client learning rate $\eta_\ell$, number of server epochs $E_g$, number of client epochs $E_\ell$, atlas size $M$, number of FL rounds $K$.
Output: model $\boldsymbol{\omega}$
1: Initialize $\mathcal{A} = \{\}$
2: repeat
3:     $j \leftarrow$ Sample available client
4:     Start the $j$th client training with $\{w^k, \eta_\ell, E_\ell\}$
5:     if receive client update then
6:         $\Delta_j \leftarrow$ Model update from the $j$th client
7:         $\{\boldsymbol{a}_m\} \leftarrow$ UpdateAtlas($\{\boldsymbol{a}_m\}, \{s_m\}, \Delta_j$)
8:     if time to start coefficient search at round $k$ then
9:         $\{\hat{c}_m\}, \boldsymbol{\omega}^{k+1} \leftarrow$ GlobalModelUpdate($\{\boldsymbol{a}_m\}, \boldsymbol{\omega}^k$)
10:        $s_m \leftarrow |\hat{c}_m|, \quad \forall m = 1, \dots, |\mathcal{A}|$.
11: until Stopped
12: function UpdateAtlas($\{\boldsymbol{a}_m\}, \{s_m\}, \Delta_j$)
13:     if $|\mathcal{A}| < M$ then
14:         $m' \leftarrow |\mathcal{A}| + 1$
15:     else if $|\mathcal{A}| == M$ then
16:         $m' = \arg\min_m \{s_m\}$
17:     $\boldsymbol{a}_{m'} \leftarrow \Delta_j$
18:     return $\{\boldsymbol{a}_m\}$
19: function GlobalModelUpdate($\{\boldsymbol{a}_m\}, \boldsymbol{\omega}^k$)
20:     $\{\bar{\boldsymbol{a}}_m\} \leftarrow$ Normalize($\{\boldsymbol{a}_m\}$) using Eq. (4)
21:     Initialize $\{c_m\}$ as $\mathbf{0}$ or with respect to Fallback
22:     $\{\hat{c}_m\} \leftarrow$ Coefficient search using Eq. (5) or Eq. (9)
23:     $\boldsymbol{\omega}^{k+1} = \boldsymbol{\omega}^k + \sum_{m=1}^{|A|} \hat{c}_m \bar{\boldsymbol{a}}_m$
24:     return $\{\hat{c}_m\}, \boldsymbol{\omega}^{k+1}$
```

This heterogeneity can lead to multiple optimization iterations during local training, causing convergence challenges for FL methods [35, 52]. To address this issue, various enhancement strategies have been proposed. One approach involves using Bayesian modeling to capture and regularize the correlation and variation between clients [10, 63, 66, 67]. Another method uses a variance reduction model to correct for client-drift in local updates [26]. Additionally, researchers have explored contrastive learning on shared representations [33] and guided local training using learned local drift [17]. A different stream of work adopts a divide-and-conquer strategy, identifying clusters of data distribution among clients to make the federated learning task more focused on the data within each cluster [18]. In some cases, the training target is even tailored to individual client's data distribution, with information shared within the FL framework used to obtain a better initialization for local adaptation [44, 50, 51, 64]. Notably, these methods do not focus on learning a global model that processes the entire data distribution.

- *Asynchronous Communication.* The previous works discussed above consider synchronous communication, where the server waits for all selected clients to complete their local training and report model updates before aggregation. However, this setup can lead to long wall-clock times due to the slowest client's downlink, local training, and uplink delays [2]. To address this issue, researchers have proposed various strategies for handling asynchronous communication. One approach involves sampling a large number of clients and aggregating model updates as soon as a minimum threshold is reached, while delayed clients are discarded [2]. However, this method can lead to skewed population data distributions observed by the server, especially when clients have inherent and consistent delay. Moreover, abandoned clients may still complete their local training, resulting in excessive energy consumption. This approach is also not applicable if client groups are not large-scale (e.g., hundreds or thousands). To mitigate these issues, some researchers have proposed reducing the local training workload of slow clients [65] or discarding model updates that exceed a predefined delay threshold (could be multiple communication rounds) [39]. However, they are not able to comprehensively address the challenges of asynchronous communication. Another line of work focuses on developing algorithms that can accept and utilize model updates regardless of their delay, allowing for more efficient asynchronous communication [31, 42, 53, 54, 58], which are the main baselines we compare with in this paper.

Alternative Decentralized Learning Schemes. Beyond FL frameworks, there are other paradigms that support decentralized training. For instance, Gossip Learning [20, 45] facilitates peer-to-peer communication among clients, eliminating the need for a central server. In this setup, distributed clients exchange and aggregate model updates directly at their local computation nodes. However, such methods typically exhibit lower efficiency compared to FL due to the lack of a centralized coordination. Given that our work focuses on hybrid data regimes where a natural center node exists, we concentrate on leveraging the FL mechanism to learn from decentralized data.

C.2. Learning from Hybrid Data

Prior studies have explored leveraging server-side data to enhance decentralized learning. These methods consider scenarios where clients lack sufficient computational resources for local training and instead upload their data to the server [15, 16, 43]. The server computes model updates on the behalf of these clients while coordinating federated learning among other clients. Additionally, work incorporating knowledge distillation into FL frameworks [32, 36, 60] leverages (collected or synthesized) server-side data to integrate clients' knowledge into the global model.

However, these approaches do not address the challenge of asynchronous communication, as knowledge is equally extracted from every client regardless of potential communication delays. In contrast, our method tackles more practical challenges and can be applied to a broader range of scenarios thanks to the accommodation of out-of-domain data availability.

Notably, Yueqi et al. [62] propose a method that searches for optimal merging coefficients, similar to our approach. However, their technique is limited to searching within the convex hull of reported models, whereas we demonstrate that optimal coefficients can be negative. Furthermore, our approach can accommodate out-of-domain data availability scenarios where server-side data is visually distinct from decentralized data, whereas these previous works are restricted to in-domain data availability.

C.3. Model Averaging

Beyond model aggregation at the server in federated learning, model averaging has been explored in other areas as well. For instance, Wortsman et al. [55] demonstrate the potential of averaging models fine-tuned with diverse hyperparameter configurations. Several works [22, 24, 59, 61] have proposed advanced averaging strategies to merge models fine-tuned for different downstream tasks in language modeling, aiming to create a new model with multiple capabilities. However, these approaches often overlook data heterogeneity, which can lead to conflicting information between different weights (or model updates), as discussed in Sec. 3. In image generation, significant oscillation has been observed during the training of diffusion models. To address this, Liu et al. [38] propose searching for optimal coefficients to merge all historical model weights. However, such averaging methods typically do not involve a cyclic optimization process between server and client, which means they also do not investigate the asynchronous communication challenge. This distinction sets our contribution apart from these related works.

D. Experimental Setting

Models. In this paper, we conduct experiments on three network architectures: **1)** A small Convolutional Neural Network (CNN), consisting of three convolutional layers, one pooling layer, and two fully connected layers. **2)** A ResNet18 [19] model pre-trained on ImageNet [13], using the checkpoint shared by PyTorch.[2] **3)** A Vision Transformer (ViT16-Base) [14] model pre-trained on ImageNet [13], using the checkpoint provided by Jia et al. [23].[3] During local training, clients update all parameters of the CNN and ResNet18 models. For ViT16-Base, we apply Low-rank Adaptation (LoRA) [21] with a rank of 4 and an adaptation scale of 8. Input images are resized to 32×32 for the CNN, 224×224 for ResNet18, and 224×224 for ViT16-Base.

[2]Source: `https://download.pytorch.org/models/resnet18-f37072fd.pth`

[3]Source: `https://github.com/KMnP/vpt`

Datasets. We employ three datasets to simulated data heterogeneity: CIFAR10/100 [29], and Fashion-MNIST [57]. For CIFAR-100, the network is trained to classify 20 superclasses, while for the other datasets, it predicts their respective 10 classes. Following previous work [63], we partition the training data among clients using a Dirichlet distribution with two concentration parameters: Dir(0.1) and Dir(0.3). For CIFAR100, we perform partitioning with respect to the 100 fine classes, which can induce label concept drift [67] and increase data heterogeneity among clients. For the other datasets, partitioning is based on their respective 10 classes. For the in-domain data availability, we reserve 1000 data points from each dataset's test set as server-side data. The remaining test data is used to evaluate the test accuracy. For the out-of-domain data availability, we use ImageNet [13] as the server-side dataset. Given that out-of-domain data often has abundant resources, we create a subset of ImageNet with 250K data points.

Additionally, we utilize two dataset containing real-world data heterogeneity: FEMNIST [12] and CelebA [40]. We partition the data by real-world individual following the LEAF framwork [5], and then randomly sample 1000 clients to form the federated learning group. For FEMNIST, we restrict the task to be predicting the first 40 classes (removing data corresponding to other classes). While for CelebA the task is to predict the "Smiling" attribute.

Methods. We compare our method `Feddle` with several competitive federated learning approaches: **1)** hybrid approaches `Fed+FT`, `HCFL` [15], `FedDF` [36], and **2)** asynchronous methods `FedAsync` [58], `FedBuff` [42], and `CA2FL` [53]. Additionally, we include `FedAvg` [41] and `Center`, which is trained exclusively on the server-side data. The learning rate and number of training epochs for the clients are tuned with Adam optimizer [27] using `FedAvg`, and these settings are subsequently applied to all other methods. Hyperparameter tuning is performed for each method using CIFAR10 as a representative dataset. The optimal hyperparameters are then applied to the other datasets, as our experiments show that they remain largely consistent across different datasets with the same network architecture.

The hyperparameters searched for each method are as follows:

- `Fed+FT`: Server learning rate $\eta \in \{1e{-}5, 1e{-}4, 1e{-}3\}$, Server epochs $M \in \{1, 10, 20\}$.

- FedDF: Server learning rate $\eta \in \{1e-5, 1e-4, 1e-3\}$, Server epochs $M \in \{1, 10, 20\}$.
- Center: Server learning rate $\eta \in \{1e-5, 1e-4, 1e-3\}$, Server epochs $M \in \{10, 20, 50\}$.
- FedAsync: Adaptive constant $a \in \{0.1, 0.5, 0.9\}$ and mixing constant $\alpha \in \{0.2, 0.4, 0.8\}$.
- FedBuff: Server learning rate $\eta \in \{0.01, 0.1, 1\}$ and buffer size $M \in \{10, 20, 50\}$.
- CA2FL: Server learning rate $\eta \in \{0.01, 0.1, 1\}$ and buffer size $M \in \{10, 20, 50\}$.
- Feddle: Server learning rate $\eta \in \{1e-5, 1e-4, 1e-3\}$, server epochs $M \in \{1, 10, 20\}$.

Model atlas size and fallback regularization strength of Feddle are determined as discussed in App. F.7. HCFL does not have additional hyperparameters as it trains a network at the server the same as the clients do.

Experimental Details. To account for client delays, we introduce staleness by sampling each client's delay from a half-normal distribution, which matches the practical distribution as observed in previous work [42]. Specifically, we take the absolute value of a sample drawn from a zero-mean Gaussian distribution and consider two standard deviations: 5 and 20. To avoid re-sampling clients that have not yet reported their model updates, each client is sampled only once until its update is received in the experiments. For a fair comparison across methods, each experiment is repeated three times with different random seeds. The random seed controls network initialization, client sampling order, client delay, and data partitioning. We evaluate the global model every 10 communication rounds and record the maximum value from the last five evaluation rounds as the final model performance. Finally, we compute the mean and standard deviation of three runs to provide a robust estimate of each method's performance.

E. Computational Efficiency

The optimization of Feddle as described in Eq. (5) requires computing gradient with respect to all anchors $\bar{\boldsymbol{a}}_1 \dots \boldsymbol{a}_{|A|} \in \mathbb{R}^d$. To enhance scalability, we reduce GPU memory usage and enable distributed computing by applying the following technique. During forward propagation (i.e. when computing Eq. (5)), we accumulate the anchors to the global model while stopping gradient propagation:

$$\boldsymbol{\omega}' = \text{stop\_grad}(\boldsymbol{\omega}^k + c_1\bar{\boldsymbol{a}}_1 + \dots c_{|\mathcal{A}|}\bar{\boldsymbol{a}}_{|\mathcal{A}|}). \quad (15)$$

During backpropagation, the gradient of the coefficients is computed as:

$$\forall m = 1, \dots, |\mathcal{A}|, \quad \partial\ell/\partial c_m = \langle \partial\ell/\partial\boldsymbol{\omega}', \bar{\boldsymbol{a}}_m \rangle. \quad (16)$$

For out-of-domain data availability, we replace ℓ with h. By employing Eqs. (15) and (16), the anchors are excluded from the gradient computation graph, allowing us to distribute the gradient calculations across multiple nodes while storing the anchors and model separately. This design enables Feddle to support large models and model atlas. However, as discussed in App. F.7, Feddle works well with a moderately sized model atlas, and its computation cost is comparable to fine-tuning a model, which is typically manageable on a server. A comparison of the computation complexity across different approaches is provided in App. F.3.

F. Additional Results

In this section, we present additional experimental results. In App. F.1 we report results on CIFAR100. In App. F.2, we show outcomes for a CNN trained from scratch. App. F.3 compares the computation cost across different approaches, while App. F.4 illustrates additional search patterns of Feddle. Finally, App. F.5 provides further analysis of the optimization signal from the surrogate loss applied in Feddle.

F.1. Experiments on CIFAR10

Table 6 summarizes the results on CIFAR10. The experimental settings align with those in Tab. 1. As observed previously, Feddle consistently outperforms the baseline methods and is less affected by high data heterogeneity and communication delays.

F.2. Training from Scratch on CNN

We supplement our results with experiments training a CNN from scratch. As shown in Tab. 7, Feddle consistently outperforms all baselines. In contrast to the results with pretrained ResNet and ViT (see Tabs. 1 and 6), we observe that Feddle may not outperform Center in the OOD setting under the most challenging scenario with Dir(0.1) and $\mathcal{N}(20)$. However, in the OOD setting Feddle often significantly outperforms the best baseline and its performance is less impacted by heterogeneity and delay. Moreover, when training a randomly initialized CNN, Feddle can leverage ImageNet data to guide the client models trained on Fashion-MNIST, further demonstrating the robustness of our framework in the OOD scenarios.

F.3. Server-Side Computation Cost

To measure the computation cost, we construct a federated learning group with 1000 clients and sample 50 clients per round. For simplicity, and to avoid the dynamics of asynchronous communication, we assume that all 50 clients report on time. We use the ViT network and fine-tune it with LoRA as described in App. D. For Feddle, the model atlas size is set to twice the number of clients sampled per round (i.e. atlas size = 100), which is good for performance as discussed in App. F.7. We also set the buffer size of

Dataset	Method	ID	ResNet18				ViT			
			Dir(0.1), $\mathcal{N}(20)$	Dir(0.1), $\mathcal{N}(5)$	Dir(0.3), $\mathcal{N}(20)$	Dir(0.3), $\mathcal{N}(5)$	Dir(0.1), $\mathcal{N}(20)$	Dir(0.1), $\mathcal{N}(5)$	Dir(0.3), $\mathcal{N}(20)$	Dir(0.3), $\mathcal{N}(5)$
CIFAR-10	Center	✓	77.3 ± 0.8				92.3 ± 0.0			
	Fed+FT	✓	78.4 ± 0.6	81.3 ± 0.2	80.1 ± 0.2	82.2 ± 0.2	96.5 ± 0.1	**97.1 ± 0.0**	96.2 ± 0.2	97.0 ± 0.0
	HFCL	✓	80.2 ± 0.9	83.7 ± 1.0	82.1 ± 0.3	85.7 ± 0.4	96.1 ± 0.1	96.9 ± 0.1	96.4 ± 0.0	96.9 ± 0.0
	FedDF-ID	✓	57.3 ± 1.5	75.4 ± 0.4	74.4 ± 1.0	82.1 ± 0.3	91.1 ± 0.7	96.8 ± 0.2	93.5 ± 0.2	96.8 ± 0.0
	Ours-ID	✓	**86.3 ± 0.4**	**86.7 ± 0.5**	**87.3 ± 0.4**	**87.8 ± 0.3**	**97.1 ± 0.0**	**97.1 ± 0.0**	**97.7 ± 0.1**	**97.5 ± 0.1**
	FedAvg		56.4 ± 5.3	75.1 ± 1.0	72.4 ± 1.0	85.2 ± 0.2	87.0 ± 1.5	94.6 ± 1.1	89.4 ± 0.1	95.5 ± 0.2
	FedAsync		65.1 ± 3.5	74.9 ± 2.4	79.3 ± 1.9	83.9 ± 1.4	89.4 ± 1.3	94.7 ± 0.8	92.8 ± 0.7	96.3 ± 0.3
	FedBuff		59.6 ± 3.3	72.8 ± 7.3	69.3 ± 8.0	77.9 ± 4.3	96.2 ± 0.1	96.1 ± 0.3	97.0 ± 0.2	96.9 ± 0.1
	CA2FL		64.4 ± 7.2	80.2 ± 0.9	71.6 ± 5.7	76.3 ± 6.8	96.5 ± 0.1	96.1 ± 0.1	97.1 ± 0.1	96.9 ± 0.0
	FedDF-OOD		29.7 ± 1.1	29.2 ± 5.6	42.9 ± 2.9	42.8 ± 2.1	29.7 ± 1.1	29.2 ± 5.6	42.9 ± 2.9	42.8 ± 2.1
	Ours-OOD		**82.2 ± 1.4**	**83.1 ± 0.3**	**86.1 ± 0.5**	**88.3 ± 0.6**	**97.0 ± 0.2**	**96.7 ± 0.4**	**97.5 ± 0.1**	**97.5 ± 0.0**

Table 6. **Comparisons of different approaches** on CIFAR10. These experiments consider two data heterogeneity levels (Dir(0.1), Dir(0.3)) and two delay levels ($\mathcal{N}(5), \mathcal{N}(20)$). "ID" indicates whether the approach uses in-domain data. If so, 1000 samples are provided. Performance higher than `Center` is underlined. The best performance in both data availabilities is highlighted by bold.

Dataset	Method	ID	CNN			
			D(0.1), $\mathcal{N}(20)$	D(0.1), $\mathcal{N}(5)$	D(0.3), $\mathcal{N}(20)$	D(0.3), $\mathcal{N}(5)$
CIFAR-10	Center	✓	42.1 ± 0.1			
	Fed+FT	✓	44.3 ± 0.5	47.4 ± 0.2	46.9 ± 0.6	51.3 ± 0.6
	HFCL	✓	44.7 ± 0.5	47.1 ± 0.4	47.6 ± 0.1	51.2 ± 1.1
	FedDF-ID	✓	18.0 ± 1.0	19.8 ± 2.0	37.3 ± 1.2	39.5 ± 1.3
	Ours-ID	✓	**51.2 ± 1.4**	**52.9 ± 0.3**	**51.5 ± 1.1**	**53.3 ± 0.6**
	FedAvg	✗	31.0 ± 1.3	35.0 ± 3.8	37.2 ± 1.2	52.0 ± 2.6
	FedAsync	✗	35.5 ± 1.4	40.4 ± 0.2	43.2 ± 0.8	45.7 ± 3.1
	FedBuff	✗	26.8 ± 3.3	33.2 ± 7.6	32.6 ± 4.1	44.0 ± 7.2
	CA2FL	✗	27.6 ± 6.9	36.6 ± 2.7	37.1 ± 3.4	43.9 ± 4.5
	FedDF-OOD	✗	18.5 ± 1.9	19.4 ± 1.9	35.1 ± 0.2	39.3 ± 1.4
	Ours-OOD	✗	**37.6 ± 4.5**	**42.3 ± 1.6**	**44.5 ± 0.8**	**53.1 ± 2.5**
CIFAR-100	Center	✓	23.3 ± 0.5			
	Fed+FT	✓	26.2 ± 0.7	29.7 ± 0.3	27.0 ± 0.3	30.3 ± 0.3
	HFCL	✓	28.9 ± 0.4	30.8 ± 0.3	29.3 ± 0.4	31.9 ± 0.5
	FedDF-ID	✓	28.7 ± 0.5	29.3 ± 0.6	28.9 ± 0.1	29.1 ± 0.2
	Ours-ID	✓	**32.7 ± 0.6**	**38.6 ± 1.6**	**38.0 ± 0.7**	**40.3 ± 0.8**
	FedAvg	✗	21.7 ± 2.2	27.7 ± 1.9	25.9 ± 1.2	32.1 ± 0.7
	FedAsync	✗	24.3 ± 1.2	30.8 ± 0.7	27.0 ± 1.1	33.0 ± 0.4
	FedBuff	✗	22.5 ± 2.2	31.5 ± 2.4	24.9 ± 4.4	31.1 ± 2.5
	CA2FL	✗	22.3 ± 0.7	29.9 ± 3.8	25.2 ± 3.2	32.0 ± 2.8
	FedDF-OOD	✗	24.1 ± 0.4	29.8 ± 0.5	27.1 ± 0.4	33.3 ± 0.9
	Ours-OOD	✗	**31.9 ± 1.8**	**37.2 ± 1.7**	**36.9 ± 1.4**	**40.4 ± 0.6**
Fashion-MNIST	Center	✓	82.6 ± 0.9			
	Fed+FT	✓	84.5 ± 0.2	86.1 ± 0.2	85.1 ± 0.0	87.1 ± 0.0
	HFCL	✓	84.4 ± 0.4	86.2 ± 0.1	85.2 ± 0.3	87.1 ± 0.2
	FedDF-ID	✓	48.9 ± 5.8	44.0 ± 5.0	72.9 ± 0.9	71.7 ± 4.9
	Ours-ID	✓	**86.5 ± 0.2**	**87.3 ± 0.0**	**86.4 ± 0.2**	**87.1 ± 0.1**
	FedAvg	✗	60.8 ± 4.4	80.7 ± 1.0	81.1 ± 0.7	86.6 ± 0.1
	FedAsync	✗	78.4 ± 2.2	80.9 ± 1.8	83.4 ± 0.6	85.4 ± 1.0
	FedBuff	✗	79.4 ± 1.0	82.4 ± 1.5	81.6 ± 4.6	85.4 ± 1.0
	CA2FL	✗	78.9 ± 0.9	85.4 ± 0.7	82.1 ± 2.7	85.5 ± 0.6
	FedDF-OOD	✗	47.4 ± 6.8	54.7 ± 5.3	71.9 ± 0.4	79.1 ± 2.6
	Ours-OOD	✗	**81.4 ± 2.4**	**86.1 ± 2.0**	**85.1 ± 0.8**	**88.2 ± 1.3**

Table 7. **Comparisons of different approaches using CNN**. These experiments consider two data heterogeneity levels (Dir(0.1), Dir(0.3)) and two delay levels ($\mathcal{N}(5), \mathcal{N}(20)$). "ID" indicates whether the approach uses in-domain data. If so, 1000 samples are provided. Performance higher than `Center` is underlined. The best performance in both data availabilities is highlighted by bold.

`FedBuff` to 100. For methods that conduct training at the server, we fix the number of iterations to be the same. In this work, we search for the optimal client training iterations based on FedAvg and apply the same setting to all methods (see App. D), ensuring that client-side computations remain identical. Therefore, our comparison focuses on the server-side computation cost. Specifically, we report two metrics that vary significantly across approaches: **1)** GFLOPs, which indicate the amount of computation performed by the server, and **2)** Cache size, which reflects the amount of intermediate results cached by each approach (note that all approaches cache the global model at a minimum).

As shown in the GFLOPs column of Tab. 8, FedAvg, FedAsync and FedBuff incur almost negligible GFLOPs at the server since they simply aggregate the model updates. The computation cost for Fed+FT and HFCL represents the cost of fine-tuning the model at the server. In comparison, Feddle requires approximately 15% more GFLOPs on the server because the gradient must be projected onto the anchors (see Eq. (16)). In contrast, FedDF demands over 10$\times$ more computation due to the inference perfomred on all reported client models for knowledge distillation.

Additionally, server must cache intermediate results during training, such as the global model. The cache size varies significantly among the approaches. As shown in the Cache (MB) column of Tab. 8, Fed+FT, HFCL, FedDF, FedAvg and FedAsync have the smallest cache size, corresponding to maintaining only the global model on the server. For these methods, all received model updates can be discarded after aggregation or training. In contrast, Feddle and FedBuff has a 35% larger cache size, which corresponds to storing 50 LoRAs saved in the model atlas or buffer. Notably, due to the cached update calibration incorporated in CA2FL, a vector of the same size as the full network is saved for each client. Consequently, the cache size of CA2FL is three orders of magnitudes larger for a total of 1000 clients, and it scales with the number of clients.

In conclusion, Feddle requires server computation similar to that of fine-tuning a model. Although it caches multiple model updates depending on the size of the model atlas, this does not significantly increase the overall cache size. While Feddle is not the most lightweight framework among the approaches, its computation cost is generally manageable for a server, and it achieves the best performance. Moreover, due to its constrained size of the model atlas, Feddle's computation complexity and cache size does not rapidly scale up with the size of the federated learning group.

F.4. Searched Coefficient Pattern

In Sec. 3, we show that model updates reported by clients often contain conflicting information due to data heterogeneity and delayed responses from asynchronous communication. Consequently, the optimal aggregation coefficients computed on the server are not always positive. Since Feddle consistently achieves the best performance by determining the aggregation coefficients under data guidance, we further demonstrate that its searched coefficients indeed include negative values. As illustrated in Fig. 8 (another subplot is provided in Fig. 6), some of the coefficients deemed optimal by the server are negative. This phenomenon persists across all communication rounds and under different configurations of data heterogeneity and communication delays. This observation highlights the persistent disagreement among clients' model updates and attests to the capability of Feddle.

Method	GFLOPs	Cache (MB)
Fed+FT	7.1×10^5	3.4×10^2
HFCL	7.1×10^5	3.4×10^2
FedDF	1.8×10^7	3.4×10^2
FedAvg	1.8×10^1	3.4×10^2
FedAsync	1.8×10^1	3.4×10^2
FedBuff	1.8×10^1	4.6×10^2
CA2FL	1.9×10^1	3.4×10^5
Feddle (Ours)	8.2×10^5	4.6×10^2

Table 8. **Comparison of the server-side computation cost across approaches.** The cost of Center is similar as HFCL and Fed+FT. The highest cost is marked in blue.

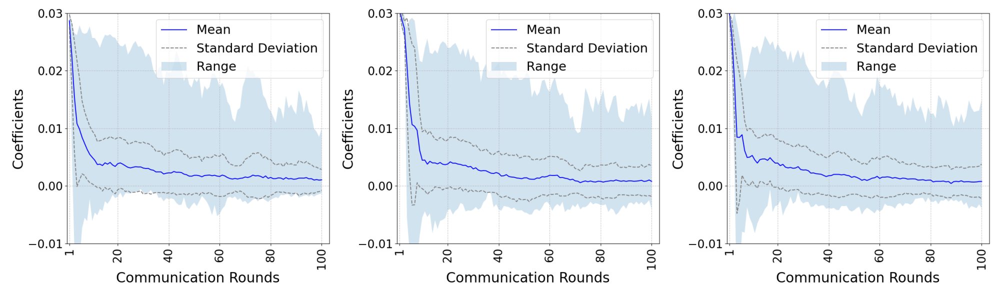
F.5. Optimization Signal of the Surrogate Loss

In Sec. 4.1.2, we discuss that if the server guides decentralized training using OOD data $\mathcal{D}'_S$ with a surrogate loss function h, the induced gradient must satisfy Eq. (7) to ensure that the optimization direction aligns with the ID server-side data $\mathcal{D}_S$ and the client loss function ℓ. For convenience, we restate this condition:

$$\langle \partial h(\mathcal{D}'_S, , \cdot)/\partial \boldsymbol{c}, ; \partial \ell(\mathcal{D}, , \cdot)/\partial \boldsymbol{c} \rangle > 0. \tag{17}$$

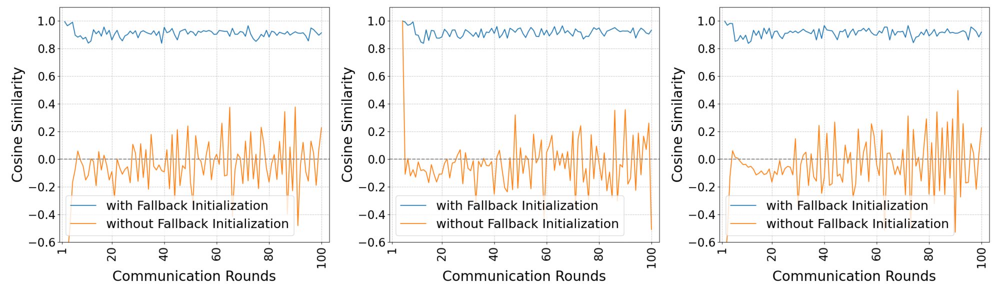
This condition is also incorporated into our theoretical analysis to justify the convergence of Feddle (see Sec. 4.2). We find that this condition is met under various settings in our experiments, and fallback initialization is crucial for its satisfaction. As shown in Fig. 9 (with an additional subplot in Fig. 7), without fallback initialization the cosine similarity between the optimization directions derived from ID and OOD data appears random. In contrast, with fallback initialization, these optimization directions become highly aligned, with the cosine similarity approaching 1. Our ablation study (see App. F.7) further confirms that without the fallback mechanism, Feddle's performance deteriorates to random guessing, underscoring the critical role of fallback initialization for applying Feddle in scenarios where only OOD data is available.

Based on these results, we hypothesize that the optimization landscape constructed by the surrogate loss using



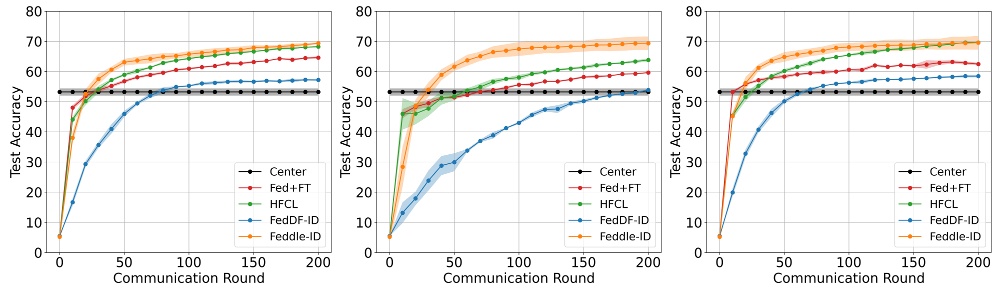
(a) $\mathrm{Dir}(0.1), \mathcal{N}(5)$ (b) $\mathrm{Dir}(0.3), \mathcal{N}(20)$ (c) $\mathrm{Dir}(0.3), \mathcal{N}(5)$

Figure 8. **Statistics of coefficients identified by `Feddle` for in-domain (ID) data availability** using ResNet18 and CIFAR-10.



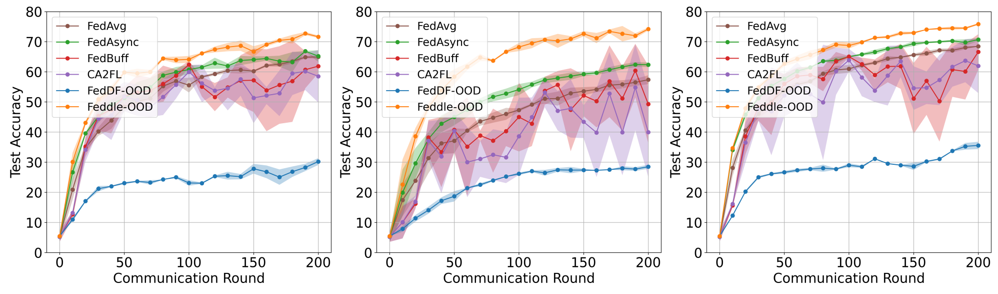
(a) $\mathrm{Dir}(0.1), \mathcal{N}(5)$ (b) $\mathrm{Dir}(0.3), \mathcal{N}(20)$ (c) $\mathrm{Dir}(0.3), \mathcal{N}(5)$

Figure 9. **Similarity of the coefficients' optimization direction between in-domain (ID) and out-of-domain (OOD) data** using ResNet18 and CIFAR-10.



(a) $\mathrm{Dir}(0.1), \mathcal{N}(5)$ (b) $\mathrm{Dir}(0.3), \mathcal{N}(20)$ (c) $\mathrm{Dir}(0.3), \mathcal{N}(5)$

Figure 10. **Convergence plots** for ResNet18 on CIFAR100 under OOD data availability.



(a) $\mathrm{Dir}(0.1), \mathcal{N}(5)$ (b) $\mathrm{Dir}(0.3), \mathcal{N}(20)$ (c) $\mathrm{Dir}(0.3), \mathcal{N}(5)$

Figure 11. **Convergence plots** for ResNet18 on CIFAR100 under ID data availability.

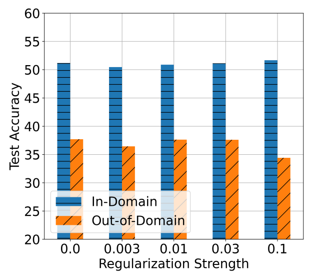


Figure 12. **Accuracy vs. regularization strength** using CNN and CIFAR10, with Dir(0.1), $\mathcal{N}(20)$.

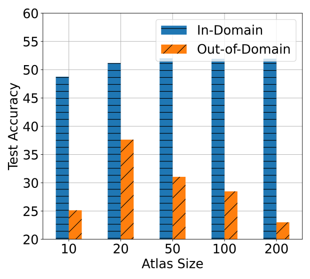


Figure 13. **Accuracy vs. atlas size** using CNN and CIFAR10, with Dir(0.1), $\mathcal{N}(20)$. *10 clients are sampled per round.*

OOD data may not be globally aligned with that formed by the client loss function using ID data. However, when the starting point is initialized near the optimum for ID data (as fallback can leverage an existing successful framework), the optimization direction derived from OOD data can still align with that of ID data.

F.6. Convergence Plots

We present the convergence plots for ResNet18 on CIFAR100 in Figs. 10 and 11. The plots corresponding to the configuration with Dir(0.1) and $\mathcal{N}(20)$ are shown in Fig. 5.

F.7. Ablation Studies

Fallback Regularization Strength. One hyperparameter introduced by `Feddle` is the regularization strength λ of the fallback mechanism. As shown in Fig. 12, with ID data, `Feddle` appears insensitive to this hyperparameter, while a strong regularization (such as $\lambda = 0.1$) may not be beneficial under the OOD data availability. In this work, we adopt 0.0 for ID settings and 0.01 for OOD settings across datasets and models.

Model Atlas Size. The other hyperparameter introduced by `Feddle` is the atlas size M. As shown in Fig. 13, expanding the model atlas generally benefits ID settings. However, an excessively large atlas, such as one that more than twice the number of clients sampled per round, can hinder performance in the OOD settings. This is likely because the optimization signal with OOD data is less reliable than ID data, making `Feddle` more prone to converging to a location that deviates from the original objective in an expanded optimization space. In this work, we always set the atlas size to twice the number of sampled clients, which results in good performance across datasets, models and settings.

Computation Cost. We compare computation cost across different approaches in terms of the server computation complexity and cache size in App. F.3.

G. Future Work

In this paper, we introduce the concept of hybrid data regimes and propose a federated dual learning framework, `Feddle`, to harness the strengths of both server-side and decentralized data. As a fundamental framework, `Feddle` opens many directions for future study and improvement. For instance, incorporating adaptive model updates, as explored in recent FL research [53, 54], could strengthen the anchors that comprise the model atlas. Additionally, the fallback mechanism can be refined for greater effectiveness, such as adaptively selecting strategies based on the delay status. Another promising direction involves leveraging unsupervised objectives to exploit unlabeled server data, thereby enriching the available data resources. Furthermore, optimizing the computational efficiency of `Feddle` through techniques such as quantization or sparsification could improve its applicability to extremely large-scale settings. For example, the anchors can be quantized or sparsified before performing the coefficient search (c.f. Eqs. (15) and (16)). Finally, integrating `Feddle` with differential privacy or trusted execution environments is an exciting area for addressing data privacy concerns. We plan to explore these avenues for improvement in future work.