FED3R: Recursive Ridge Regression for Federated Learning with strong pre-trained models

Eros Fanì
Politecnico di Torino
eros.fani@polito.it

Raffaello Camoriano
Politecnico di Torino
raffaello.camoriano@polito.it

Barbara Caputo
Politecnico di Torino
barbara.caputo@polito.it

Marco Ciccone
Politecnico di Torino
marco.ciccone@polito.it

Abstract

Current Federated Learning (FL) methods often struggle with high statistical heterogeneity across clients' data, resulting in client drift due to biased local solutions. This issue is particularly pronounced in the final classification layer, negatively impacting convergence speed and accuracy throughout model aggregation. To overcome these challenges, we introduce Federated Recursive Ridge Regression (FED3R). Our method replaces the softmax classifier with a ridge regression-based one computed in a closed form, ensuring robustness to statistical heterogeneity and drastically reducing convergence time and communication costs. When the feature extractor is fixed, the incremental formulation of FED3R is equivalent to the exact centralized solution. Thus, Fed3R enables higher-capacity pre-trained feature extractors with better predictive performance, incompatible with previous FL techniques, because no backpropagation is required through the feature extractor, and only a few rounds are needed to converge. We propose FED3R in three variants, with FED3R-RF significantly enhancing performance to levels akin to centralized training while remaining competitive regarding the total communication costs.

1 Introduction

Federated Learning (FL) offers a practical framework to train machine learning models collaboratively across *clients* while ensuring that data never leaves the devices, hence mitigating potential privacy risks. In its simplest version, FedAvg [1], federated training involves multiple communication rounds between clients and server. Each client optimizes their local models independently on their private data, sends the parameters to a central server that aggregates them, and transmits the updated model to the next clients. While appealing, limiting loss minimization to local datasets presents several challenges. In real-world scenarios, the number of clients can reach billions [2], and data are collected based on user preferences, availability, geographical location [3, 4, 5, 6], or personal habits [7, 8]. This leads to data distributions across clients with inherent *statistical heterogeneity* in the form of *quantity skewness* [9, 10], *label skewness* [11, 12, 13], or *domain shift* [5, 4].

As a result, training models that generalize well across the global underlying data distribution pose a significant challenge. In particular, convergence speed is hampered due to clients' sparse sampling and partial participation [14, 15]. Furthermore, biased updates from individual clients can cause the model to deviate from its global convergence points [16, 17, 18].

International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023 (FL@FM-NeurIPS'23). This workshop does not have official proceedings and this paper is non-archival.

Most of the approaches addressing these issues focus on client-side training by regularizing the local objective to reduce *client drift* [17, 16, 18, 19] or leveraging momentum to incorporate knowledge from previous updates and align the local optimization to the global direction [15, 20, 21, 22].

The authors of [23] show that deeper layers in Neural Networks are more susceptible to bias toward the individual client data distribution, while initial layers maintain better consistency. Similarly to Continual Learning [24, 25], this phenomenon occurs due to the data distribution being non-i.i.d. and the nature of the softmax-based classifier which, once updated, is prone to forgetting past experience [26]. Indeed, local optimization leads to biased solutions towards local data, and users may not be revisited during training [27, 12]. See Appendix A for more related works.

Contributions. Following a recent trend in Federated and Continual Learning [28, 29, 30] leveraging pre-trained representations, we propose Federated Recursive Ridge Regression (FED3R). FED3R employs a Ridge Regression (RR) classifier [31] in federated settings to tackle the issues of statistical heterogeneity. We show how the exact centralized closed-form solution of the RR problem can be computed collaboratively by the clients by repurposing the recursive RR formulation to the FL setting to design a classifier immune to client drift, catastrophic forgetting, and statistical heterogeneity. Indeed, by fixing the representation, the global solution can be computed incrementally without gradient-based updates, resulting in an algorithm invariant to the data split across clients and the order in which they are sampled during training. To address the distribution shift of the target task from the pre-trained representation, we also introduce FED3R with Fine-Tuning (FED3R-FT), which can update the feature extractor while efficiently computing the RR classifier. In addition, we propose a kernelized version of FED3R based on Random Features (RF) (FED3R-RF) that allows trading off computational and communication costs with improved predictive performance. Nevertheless, these costs are still competitive with other methods specifically developed for statistical heterogeneity as the number of rounds for convergence is significantly reduced.

We empirically evaluate the effectiveness of our proposed algorithms on the Landmarks-Users-160K dataset [3], a realistic FL scenario for visual classification with thousands of clients and pronounced statistical heterogeneity converging up to 44$\times$ and 18$\times$ times faster than FedAvg and Scaffold [16].

2 Background

In this section, we provide a concise overview of the Federated Learning framework and the fundamental concepts of RR, before formally describing our algorithm.

2.1 FL problem formulation

Let $\mathcal{K}$ be the set of all the clients involved in the training forming the federation of cardinality $|\mathcal{K}| = K$, and let $\mathcal{S}$ be the server that orchestrates the training procedure. Each client $k \in \mathcal{K}$ has access to a private local dataset $\mathcal{D}_k$ of size $n_k = |\mathcal{D}_k|$. The server and other clients do not have access to individual client data of the clients. Each local dataset $\mathcal{D}_k$ is composed of n_k pairs $(x, y) \sim P_k$, where $x \in \mathcal{X}, y \in \mathcal{Y}$, $\mathcal{X}$ and $\mathcal{Y}$ are the input and output space respectively, and P_k is the joint distribution associated with the client k. In the case of the RGB image class, $\mathcal{X} = \mathbb{R}^{3\times W\times H}$ and $\mathcal{Y} = \{0, 1\}^C$, where C is a fixed number of classes.

Therefore, the global federated objective is given by $\theta^* = \arg\min_\theta \sum_{k\in\mathcal{K}} \mathcal{L}_k(\mathcal{M}(\theta); \mathcal{D}_k)$, where $\mathcal{L}_k = \sum_{(x,y)\in\mathcal{D}_k} \mathcal{L}(\mathcal{M}(x; \theta), y)$ is the local empirical risk computed according to a loss function $\mathcal{L}$ (e.g., cross-entropy) associated to the client k and $\mathcal{M}$ is a model parameterized by θ. At each round t, a subset of selected clients $\mathcal{K}' \subseteq \mathcal{K}$ train their local models on private datasets $\mathcal{D}_k$ starting from the same θ^{t-1} initialization. Then, the locally optimized model parameters are shared with the server $\mathcal{S}$, which aggregates them according to the specific FL algorithm. For instance, the FedAvg [1] aggregation rule is a weighted average of clients' models $\theta^t = \sum_{k\in\mathcal{K}'} \frac{n_k}{n} \theta_k^t$, where $n = \sum_{k\in\mathcal{K}} n_k$. The server broadcasts the aggregated model θ^t to the new active clients and the process is repeated for several rounds until convergence.

2.2 Ridge Regression (RR)

Consider the Ridge Regression problem (or Tikhonov Regularized Least Squares problem) [32] $\arg\min_{f\in\mathcal{F}} \mathbb{E}_{(x,y)\sim P_{\mathcal{X},\mathcal{Y}}} \left[(f(x)-y)^2 + \lambda \left\|f(x)\right\|_2^2\right]$, where $\mathcal{F}$ is the set of all the possible hypotheses, $P_{\mathcal{X},\mathcal{Y}}$ is the joint distribution of the input-output pairs and $\lambda > 0$.

If we restrict the hypothesis f to all the possible linear hyperplanes $f(x) = xW$, where $W \in \mathbb{R}^{d\times C}$, we can reformulate the problem as follows:

$$\underset{W\in\mathbb{R}^{d\times C}}{\arg\min} \sum_{(x,y)\in\mathcal{D}} \left\|Y - XW\right\|_2^2 + \lambda \left\|W\right\|_2^2, \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ and $Y \in \mathbb{R}^{n\times C}$ are the matrices of stacked input samples x and one-hot encoded ground-truth outputs y respectively, where n is the number of samples in the dataset $\mathcal{D}$ such that $(x,y) \sim P_{\mathcal{X},\mathcal{Y}}\ \forall (x,y) \in \mathcal{D}$ and d is the dimensionality of the input features. A closed-form optimal solution W^* to 1 exists, and is the solution of the following linear system:

$$(A + \lambda I_d)W = b\Gamma^{\alpha}, \tag{2}$$

where $A = X^\top X \in \mathbb{R}^{d\times d}$, $b = X^\top Y \in \mathbb{R}^{d\times C}$, I_d is the identity matrix of shape $d \times d$, and $\Gamma \in \mathbb{R}^{C\times C}$ is a rebalancing diagonal matrix to handle class unbalanced distributions [33], whose entries correspond to the inverse frequencies of the classes with respect to n, and $\alpha > 0$ is a label recording hyper-parameter. Additional details on the rebalancing matrix Γ are in Appendix B.

2.3 Recursive Ridge Regression Formulation

If a new labeled sample becomes available, the RR statistics can be efficiently and exactly updated without requiring costly re-training from scratch on the entire dataset [34, 35, 36]:

$$A_{n+1} = A_n + x_{n+1}x_{n+1}^\top; \quad b_{n+1} = b_n + x_{n+1}e_{y_{n+1}}^\top, \tag{3}$$

where (x_{n+1}, y_{n+1}) is the new available pair and $e_{y_{n+1}} \in \mathbb{R}^C$ is the one-hot encoding for class y_{n+1}.

Now, we can generalize Eq. 3 to the case where a set $\mathcal{D}_m$ of m new pairs is observed:

$$A_{n+m} = A_n + \sum_{(x,y)\in\mathcal{D}_m} xx^\top; \quad b_{n+m} = b_n + \sum_{(x,y)\in\mathcal{D}_m} xe_y^\top \tag{4}$$

and obtain the final RR classifier by substituting A_{n+m} and b_{n+m} in Equation 2.

3 Methods

Drawing from the previous section, we can introduce the FED3R algorithm by repurposing the recursive RR formula to the FL setting where each client, once sampled, introduces a new set $\mathcal{D}_k$ consisting of n_k pairs (x, y).

We also leverage a pre-trained representation as the input space for the RR algorithm rather than the original input space. Subsequently, we introduce the FED3R-FT variant, where we allow for fine-tuning the feature extractor across rounds using a gradient-based FL method. Finally, we present FED3R-RF, a kernelized version that uses random features [37] to approximate the Kernel Ridge Regression (KRR) solution.

3.1 Federated Recursive Ridge Regression (FED3R)

Let $\varphi : \mathcal{X} \to \mathcal{Z}$ be the pre-trained feature extractor of a model $\mathcal{M}$, mapping the input space $\mathcal{X}$ onto the latent feature space $\mathcal{Z} \subseteq \mathbb{R}^d$, and $\phi : \mathcal{Z} \to \mathcal{Y}$ be the classifier, mapping the latent feature space to the output space $\mathcal{Y}$. Given a client k and all the pairs $(x_i, y_i) \in \mathcal{D}_k$, we define X_k as the tensor of all the n_k stacked input samples x_i, and $Y_k \in \{0,1\}^{n_k\times C}$ as the matrix of corresponding one-hot encoded vectors for all the y_i.

Algorithm 1: FED3R

Require:
Server $\mathcal{S}$, clients $k \in \mathcal{K}$, local datasets $\mathcal{D}_k$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$
Hyper-parameters $\lambda > 0$, $\alpha > 0$
for *each* *client k in any order* **do**
  $Z_k = \varphi(X_k)$, $A_k = Z_k^\top Z_k$, $b_k = Z_k^\top Y_k$
  $g_k = Y_k^\top \mathbf{1}$, $\mathbf{1} \in \mathbb{R}^{n_k}$
$\mathcal{S}$ collects all the clients statistics and computes:
$A = \sum_{k\in\mathcal{K}} A_k$, $b = \sum_{k\in\mathcal{K}} b_k$
$g = \sum_{k\in\mathcal{K}} g_k$, $\Gamma = diag\left(C\frac{\|g\|_1}{g}\right)$
Solve Eq. 2 to get W^*

As soon as each client k is idle and the server is ready to receive statistics, the clients activate. At this point, k must have access to the pre-trained feature extractor parameters θ_φ. There are two possible equivalent scenarios: either the clients have been deployed with these parameters before, or the server communicates them promptly as needed. Once active, the client k extracts the local feature maps $Z_k = \varphi(X_k)^1 \in \mathbb{R}^{n_k \times d}$, where $n_k = |\mathcal{D}_k|$, and communicates $A_k = Z_k^\top Z_k$, $b_k = Z_k^\top Y_k$ and the local number of samples per class g_k to the server. After the server has received these statistics from all the clients, it calculates the RR matrices $A = \sum_{k \in \mathcal{K}} A_k$ and $b = \sum_{k \in \mathcal{K}} b_k$, and solves the linear system of Eq. 2.

Invariance to clients split. Note that, when all the clients have been sampled, this solution is equivalent to the corresponding optimal RR solution of the centralized scenario where $\mathcal{D} = \bigcup_{k \in \mathcal{K}} \mathcal{D}_k$. In particular, the RR solution is invariant to the clients split of the centralized dataset, thanks to the permutation invariance of the summation operator [29]:

$$A = \sum_{(x,y) \in \mathcal{D}} \varphi(x)\varphi(x)^\top = \sum_{k \in \mathcal{K}} \sum_{(x,y) \in \mathcal{D}_k} \varphi(x)\varphi(x)^\top \tag{5}$$

$$b = \sum_{(x,y) \in \mathcal{D}} \varphi(x)e_y^\top = \sum_{k \in \mathcal{K}} \sum_{(x,y) \in \mathcal{D}_k} \varphi(x)e_y^\top \tag{6}$$

Finally, note that each client has to communicate its updates only once, compared to the classical gradient-based FL algorithms that require multiple training rounds.

See Appendix B for a detailed explanation of why we construct the rebalancing matrix Γ as described in Algorithm 1, and Appendix D for the privacy implications of FED3R.

3.2 Federated Recursive Ridge Regression with Fine-Tuning (FED3R-FT)

As pre-trained feature extractors may not provide discriminative enough feature maps if the target task is too dissimilar or drifts from the pre-training one, we propose to fine-tune the feature extractor through a gradient-based FL algorithm, guiding the training via a distinct softmax classifier $\phi(x; \theta_\phi)$, while computing the RR statistics. However, the RR statistics need to be adjusted to counteract the features drift over time.

As in standard FL, a group of clients $\mathcal{K}'$ is sampled with replacement during each round. Therefore, each client k is potentially sampled again at different rounds, communicating statistics computed using different versions of the feature extractor. To update the global RR statistics, we make the reasonable assumption that the representation smoothly varies during training and let each client k have an internal state $(\tilde{A}_k^{\text{old}}, \tilde{b}_k^{\text{old}}, s_k)$, to store statistics extracted at previous rounds and a flag $s_k \in \{0, 1\}$ indicating if the class counting vector g_k, needed to compute the Γ matrix, has already been communicated to the server once.

In addition, the server $\mathcal{S}$ has a global state (A^t, b^t, g^t), needed to incrementally aggregate the statistics received from the clients after each round t.

FED3R-FT Algorithm. During round t, each sampled client computes Z_k^t using the feature extractor $\varphi(x; \theta_\varphi^{t-1})$ before computing the updated feature extractor parameters $\theta_{\varphi,k}^t$. Then, it employs Z_k^t to compute the new statistics $\tilde{A}_k^{\text{new}} = (Z_k^t)^\top Z_k^t$ and $\tilde{b}_k^{\text{new}} = (Z_k^t)^\top Y_k$, and communicates $\tilde{A}_k^t = \tilde{A}_k^{\text{new}} - \tilde{A}_k^{\text{old}}$, $\tilde{b}_k^t = \tilde{b}_k^{\text{new}} - \tilde{b}_k^{\text{old}}$ and $g_k = (1 - s_k)Y_k^\top \mathbf{1}$. Thereafter, the sampled clients and the server update their internal states:

$$\text{Client } k\text{: } (\tilde{A}_k^{\text{old}}, \tilde{b}_k^{\text{old}}, s_k) \leftarrow (\tilde{A}_k^{\text{new}}, \tilde{b}_k^{\text{new}}, 1), \tag{7}$$

$$\text{Server } \mathcal{S}\text{: } (A^t, b^t, g^t) \leftarrow \left(A^{t-1} + \sum_{k \in \mathcal{K}'} \tilde{A}_k^t,\ b^{t-1} + \sum_{k \in \mathcal{K}'} \tilde{b}_k^t,\ g^{t-1} + \sum_{k \in \mathcal{K}'} g_k \right). \tag{8}$$

Finally, the server may update the rebalancing matrix Γ as illustrated in Section 3.1, and solve Eq. 2 to obtain the RR classifier.

[1] With a slight abuse of notation, here we mean the matrix of all the stacked feature maps $\varphi(x)$ computed from each input sample $x : (x, y) \in \mathcal{D}_k$.

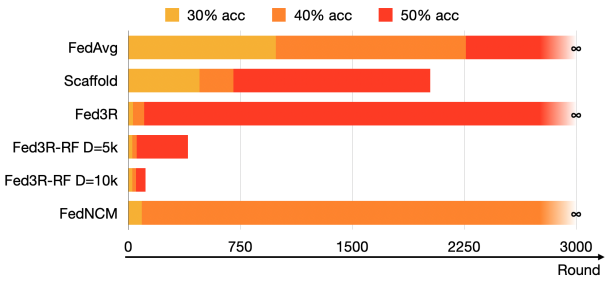


Figure 1: Rounds required to reach the target accuracy for the experiments with fixed feature extractor.

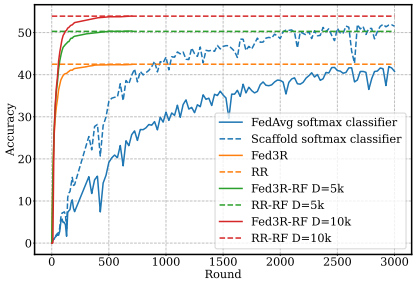


Figure 2: Comparison with FedAvg and Scaffold, with fixed feature extractor.

FED3R-FT lets the latent feature space change between classifier updates. Therefore, the solution provided by Eq. 2 is sub-optimal. However, as we show in Section 4, this issue is negligible compared to the benefits of updating the classifier based on more expressive features. The full algorithm and additional details on FED3R-FT can be found in Appendix C.

3.3 FED3R with Random Features approximation for Kernel RR (FED3R-RF)

Even if the feature extractor is trained for the target task as in FED3R-FT, it may still not be expressive enough to linearly separate the feature space. To solve this issue, we propose to employ Kernel Ridge Regression (KRR), a nonparametric learning algorithm using kernel functions, to implicitly handle non-linearity [38, 39]. However, the kernel matrix's space complexity is $\mathcal{O}(n^2)$, in contrast to the covariance matrix A with a space complexity of $\mathcal{O}(d^2)$. For sizable datasets, computing the exact KRR solution becomes impractical. Data-dependent sub-sampling methods [40, 41, 42] proved to be successful in scaling up kernel machines. However, adapting them to FL is non-trivial. In this work we focus on a data-independent method, Random Features KRR [37]. Its properties are particularly suitable for the FL setting, rendering KRR suitable for large datasets and enabling the use of the same linear RR formulation from Section 2.2.

Therefore, we propose to use a random features approximation of the RBF Kernel $k(z, \zeta) = e^{-\|z-\zeta\|^2/2\sigma^2}$, $z, \zeta \in \mathbb{R}^d$ both with FED3R and FED3R-FT. The algorithms are equivalent, with the sole exception that we now map the features z, ζ of the latent space $\mathcal{Z}$ to the latent space $\mathcal{Z}' \subseteq \mathbb{R}^D$ using random features, where D is a hyper-parameter controlling the number of random features for the approximation. Consequently, all the dimensionalities of the statistics that depended on d here depend on $D > d$.

4 Experiments

In the following, we compare the performance and communication costs between our algorithms and FedAvg [1], Scaffold [16] and FedNCM [28]. If not specified differently, we train a MobileNetV2 [43] architecture. We use the Landmarks-Users-160K dataset [44] partitioned by [3] for FL in all the experiments. For more details on the implementation, you can refer to Appendix E. Moreover, Appendix F provides additional information on how the communication costs have been estimated. Further experiments investigating the performances of the random features approximation and the impact of the number of clients sampled per round are detailed in Appendix H and Appendix I.

Experiments with fixed feature extractor. In this section, we evaluate FED3R against the baselines with a fixed feature extractor. Figure 1 shows how much FED3R and FED3R-RF are faster in reaching the target accuracies compared to the other methods. However, in the long run, Scaffold can recover and provide performance comparable with FED3R-RF $D = 5k$, as shown in Figure 2. The RR curve represents the centralized method described in Section 2.2, which is the upper bound equivalent to the case when all clients are sampled in each round. FED3R converges exactly to the RR performance after a small number of rounds, as expected (see Appendix I). Furthermore, in Table 1, we compare the communication costs required by the algorithms considering the same uniform client sampling strategy for all the methods and the final accuracies they achieve at round 3000.

Given the non-linear separability of the feature space, we also employ FED3R-RF, which consistently delivers excellent results, surpassing Scaffold and FedAvg in terms of accuracy by approximately

Table 1: Communication costs and final performance for the experiments with fixed feature extractor.

Algorithm	Downstr./k	Upstream/k	Round/k	Downstream	Upstream	Round	30% acc	40% acc	50% acc	Acc r=3k (%)
FedAvg	12.00 MB	12.00 MB	24.00 MB	120.00 MB	120.00 MB	240.00 MB	237.36 GB	542.40 GB	-	40.79
Scaffold	24.00 MB	24.00 MB	48.00 MB	240.00 MB	240.00 MB	480.00 MB	228.48 GB	337.44 GB	970.56 GB	51.52
FED3R	0.00 B	16.95 MB	16.95 MB	0.00 B	169.45 MB	169.45 MB	**5.42 GB**	**17.96 GB**	-	42.47
FED3R-RF 5k	0.00 B	140.57 MB	140.57 MB	0.00 B	1.41 GB	1.41 GB	37.95 GB	78.72 GB	560.87 GB	50.4
FED3R-RF 10k	0.00 B	481.13 MB	481.13 MB	0.00 B	4.81 GB	4.81 GB	129.90 GB	250.19 GB	**553.30 GB**	**53.96**
FedNCM	0.00 B	10.38 MB	10.38 MB	0.00 B	103.83 MB	103.83 MB	9.35 GB	-	-	35.93

2% and 13% while also demonstrating significantly faster convergence. In particular, FED3R-RF $D = 10k$ is 18× and 44× faster than Scaffold and FedAvg to reach 50% and 40% accuracy, respectively. Although FED3R-RF incurs a higher communication cost per client, those are warranted when evaluating them in terms of total communication required to reach a target accuracy. However, when the communication costs per client are a constraint, we observe that FED3R still outperforms FedNCM and FedAvg, the two least communication expensive methods together with FED3R, by 7% and 2% in terms of accuracy. Finally, it is worth noting that at least one method from the FED3R family consistently outperforms all other approaches in terms of (less) communication costs and higher accuracy.

Experiments with fine-tuned feature extractor. In this section, we evaluate the performance of FED3R-FT when the feature extractor is fine-tuned using a standard gradient-based optimization algorithm.

Here, we empirically show the advantages of introducing feature extractor fine-tuning. In this case, the communication costs of FED3R-FT are added to the costs of the underlying method for the fine-tuning (e.g. FedAvg or Scaffold). Results are shown in Table 2. Similarly to Section 4, FED3R-FT and FED3R-RF are significantly faster than all the other algorithms. Eventually, Scaffold is the only one able to reach comparable accuracy to FED3R-RF at convergence. Appendix G.1 provides further details on these experiments, including the communication costs evaluation. Moreover, Appendix J also shows a possible technique to mitigate the effect of the feature extractor shift when computing the RR statistics with FED3R-RF.

Table 2: Rounds to reach the target accuracy with fine-tuned feature extractor and final performance.

Algorithm	40% acc	50% acc	60% acc	Acc r=3k (%)
FedAvg	500	999	-	58.17
Scaffold	281	484	1038	62.32
FED3R-FT FedAvg	172	774	-	52.95
FED3R-FT Scaffold	168	547	-	54.02
FED3R-RF $D = 5k$ FedAvg	95	321	-	58.60
FED3R-RF $D = 5k$ Scaffold	98	256	1077	60.11
FED3R-RF $D = 10k$ FedAvg	**86**	212	1077	61.61
FED3R-RF $D = 10k$ Scaffold	88	**178**	**659**	**62.69**
FedNCM FedAvg	420	2038	-	50.70
FedNCM Scaffold	261	818	-	53.49

Enabling Foundation Models in FL. Gradient-based FL algorithms require lightweight networks due to the clients' limited computational capabilities. However, we can mitigate computational costs by maintaining the feature extractor parameters fixed, thus conducting only forward passes. This approach enables the use of more complex architectures with superior generalization capabilities in the context of FL. We employ this strategy also in FED3R. In Table 3, the number of rounds required to achieve the target accuracy and the final performance are presented for different feature extractor architectures. Once again, FED3R-FT and FED3R-RF demonstrate significantly faster convergence. For comprehensive results and communication costs, please refer to Appendix G.2, highlighting how our methods consistently yield substantial savings in total communication costs.

Table 3: Rounds to reach the target accuracy and final performance with more complex architectures.

Architecture	Algorithm	60% acc	70% acc	80% acc	Acc r=3k (%)
ViT [45]	FedAvg	314	539	2168	81.23
	FED3R	29	122	-	72.11
	FED3R-RF $D = 10k$	**23**	**39**	**165**	**81.49**
SWIN [46]	FedAvg	937	2439	-	71.03
	FED3R	61	-	-	65.35
	FED3R-RF $D = 10k$	**42**	**124**	-	**73.17**
ConvNext [47]	FedAvg	1424	-	-	66.92
	FED3R	120	-	-	62.69
	FED3R-RF $D = 10k$	**46**	**219**	-	**71.43**

5 Conclusion

In this work, we propose FED3R, an FL algorithm based on Recursive Ridge Regression. The design of FED3R is geared towards minimizing communication costs and accelerating convergence time while adhering to the privacy constraints inherent in FL. We propose our method in three variants, permitting feature extractor fine-tuning (FED3R-FT) and Kernel Ridge Regression (KRR) approximation via random features (FED3R-RF). FED3R efficiency opens avenues for employing more complex architectures in FL, achieving performance levels previously challenging with foundation models and more powerful architectures. Future works may extend this work to incremental learning or personalized learning scenarios within the FL framework.

Acknowledgments and Disclosure of Funding

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer, 2020, pp. 76–92.

[4] L. Fantauzzo, E. Fanì, D. Caldarola, A. Tavera, F. Cermelli, M. Ciccone, and B. Caputo, "Feddrive: Generalizing federated learning to semantic segmentation in autonomous driving," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 504–11 511.

[5] D. Shenaj, E. Fanì, M. Toldo, D. Caldarola, A. Tavera, U. Michieli, M. Ciccone, P. Zanuttigh, and B. Caputo, "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 444–454.

[6] J. Miao, Z. Yang, L. Fan, and Y. Yang, "Fedseg: Class-heterogeneous federated learning for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8042–8052.

[7] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *Advances in Neural Information Processing Systems*, 2020.

[8] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[9] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets," *arXiv preprint arXiv:2008.03371*, 2020.

[10] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[11] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[12] D. Caldarola, B. Caputo, and M. Ciccone, "Improving generalization in federated learning by seeking flat minima," in *European Conference on Computer Vision*. Springer, 2022, pp. 654–672.

[13] E. Fanì, M. Ciccone, and B. Caputo, "Feddrive v2: an analysis of the impact of label skewness in federated semantic segmentation for autonomous driving," *5th Italian Conference on Robotics and Intelligent Machines (I-RIM)*, 2023.

[14] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *International Conference on Learning Representations*, 2020.

[15] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," *Advances in Neural Information Processing Systems*, 2020.

[16] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.

[17] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[18] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *International Conference on Learning Representations*, 2021.

[19] E. Ozfatura, K. Ozfatura, and D. Gündüz, "Fedadc: Accelerated federated learning with drift control," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 467–472.

[20] J. Xu, S. Wang, L. Wang, and A. C.-C. Yao, "Fedcm: Federated learning with client-level momentum," *arXiv preprint arXiv:2106.10874*, 2021.

[21] G. Kim, J. Kim, and B. Han, "Communication-efficient federated learning with acceleration of global momentum," *arXiv preprint arXiv:2201.03172*, 2022.

[22] Y. Liu, Y. Sun, Z. Ding, L. Shen, B. Liu, and D. Tao, "Enhance local consistency in federated learning: A multi-step inertial momentum approach," *arXiv preprint arXiv:2302.05726*, 2023.

[23] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.

[24] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[25] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[27] G. Legate, L. Caccia, and E. Belilovsky, "Re-weighted softmax cross-entropy to control forgetting in federated learning," *arXiv preprint arXiv:2304.05260*, 2023.

[28] G. Legate, N. Bernier, L. Caccia, E. Oyallon, and E. Belilovsky, "Guiding the last layer in federated learning with pre-trained models," *arXiv preprint arXiv:2306.03937*, 2023.

[29] R. Wang, M. Ciccone, G. Luise, M. Pontil, A. Yapp, and C. Ciliberto, "Schedule-robust online continual learning," *arXiv preprint arXiv:2210.05561*, 2022.

[30] J. Nguyen, K. Malik, M. Sanjabi, and M. Rabbat, "Where to begin? exploring the impact of pre-training and initialization in federated learning," *International Conference on Learning Representations*, 2023.

[31] R. Rifkin, G. Yeo, T. Poggio *et al.*, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.

[32] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[33] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental robot learning of new objects with fixed update time," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3207–3214.

[34] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear estimation*. Prentice Hall, 2000, no. BOOK.

[35] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[36] A. H. Sayed, *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[37] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[38] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[39] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[40] A. Rudi, L. Carratino, and L. Rosasco, "Falkon: An optimal large scale kernel method," *Advances in neural information processing systems*, vol. 30, 2017.

[41] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," *Advances in neural information processing systems*, vol. 13, 2000.

[42] A. Rudi, R. Camoriano, and L. Rosasco, "Less is more: Nyström computational regularization," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[44] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2575–2584.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[46] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[47] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[48] T. Zhou, J. Zhang, and D. Tsang, "Fedfa: federated learning with feature anchors to align feature and classifier for heterogeneous data," *arXiv preprint arXiv:2211.09299*, 2022.

[49] Z. Li, X. Shang, R. He, T. Lin, and C. Wu, "No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier," *arXiv preprint arXiv:2303.10058*, 2023.

[50] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "Feddc: Federated learning with non-iid data via local drift decoupling and correction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 112–10 121.

[51] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, "Local learning matters: Rethinking data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 8397–8406.

[52] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[53] A. Hilmkil, S. Callh, M. Barbieri, L. R. Sütfeld, E. L. Zec, and O. Mogren, "Scaling federated learning for fine-tuning of large language models," in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2021, pp. 15–23.

[54] Z. Liu, D. Li, J. Fernandez-Marques, S. Laskaridis, Y. Gao, Ł. Dudziak, S. Z. Li, S. X. Hu, and T. Hospedales, "Federated learning for inference at anytime and anywhere," *arXiv preprint arXiv:2212.04084*, 2022.

[55] G. Sun, M. Mendieta, T. Yang, and C. Chen, "Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning," 2023. [Online]. Available: https://openreview.net/forum?id=EmH1WE1fRbt

[56] X. Zhang, M. Li, X. Chang, J. Chen, A. K. Roy-Chowdhury, A. T. Suresh, and S. Oymak, "Fedyolo: Augmenting federated learning with pretrained transformers," *arXiv preprint arXiv:2307.04905*, 2023.

[57] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[58] W. Lu, X. Hu, J. Wang, and X. Xie, "Fedclip: Fast generalization and personalization for clip in federated learning," *arXiv preprint arXiv:2302.13485*, 2023.

[59] Y. Yu, A. Wei, S. P. Karimireddy, Y. Ma, and M. Jordan, "Tct: Convexifying federated learning using bootstrapped neural tangent kernels," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 882–30 897, 2022.

[60] A. Afonin and S. P. Karimireddy, "Towards model agnostic federated learning using knowledge distillation," *arXiv preprint arXiv:2110.15210*, 2021.

[61] J. Cai, X. Liu, Z. Yu, K. Guo, and J. Li, "Efficient vertical federated learning method for ridge regression of large-scale samples," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[62] L. Huang, Z. Li, J. Sun, and H. Zhao, "Coresets for vertical federated learning: Regularized linear regression and k-means clustering," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 566–29 581, 2022.

[63] Y. Zhang, J. Duchi, and M. Wainwright, "Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3299–3340, 2015.

[64] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2657–2681, 2016.

[65] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[66] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *International Conference on Learning Representations*, 2021.

[67] R. Wightman, "Pytorch image models," https://github.com/huggingface/pytorch-image-models, 2019.

[68] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[69] W. Stadje, "The collector's problem with group drawings," *Advances in Applied Probability*, vol. 22, no. 4, pp. 866–882, 1990.

[70] M. Ferrante and N. Frigo, "A note on the coupon-collector's problem with multiple arrivals and the random sampling," *arXiv preprint arXiv:1209.2667*, 2012.

[71] M. Ferrante and M. Saltalamacchia, "The coupon collector's problem," *Materials matemàtics*, pp. 0001–35, 2014.

[72] D. Caldarola, B. Caputo, M. Ciccone *et al.*, "Window-based model averaging improves generalization in heterogeneous federated learning," in *Women in Computer Vision Workshop*. CVF/IEEE Computer Society, 2023.

Appendix

A Additional detail on the Related Works

Classifier bias and destructive interference. In strong statistically heterogeneous scenarios, the local updates of individual clients during any given training round may not be consistent. This is because the landscapes of the local loss may vary significantly. As a consequence, client drift happens, which negatively impacts convergence time [12]. The authors of [23] show that the bias of the clients is more pronounced in deeper layers of a neural network, with the peak in the deepest layer i.e. the classifier. Moreover, prior works observed that biased classifiers and misaligned features create a vicious cycle [48, 49]. Finally, the classifier bias phenomenon causes destructive interference during the aggregation phase at the end of each round, further slowing and affecting the training procedure.

Partial solutions in the literature mitigate the classifier bias problem by reducing the gap between the global model and the local ones by modifying the loss function [50], or rethinking the statistical heterogeneity problem as a generalization problem as in the standard centralized setting [51]. Following the latter direction, FedNCM [28] fits a classifier using Nearest Class Means (NCM). FedNCM is a simple method that is not gradient-based in its first phase. However, contrary to FED3R, it requires a gradient-based second phase, where they fine-tune the whole model starting from the powerful classification head derived in the first phase using NCM.

Impact of pre-training. Communication is one of the fundamental bottlenecks of FL from both timing and energy consumption perspectives [52]. Therefore, using a lightweight network is often necessary when training from scratch. However, the possibility of using a pre-trained model to fine-tune only deeper layers may allow the usage of some more complex architectures [53] because of reduced computation and communication requirements since there is no need to propagate the gradients through the fixed shallower layers and only the deeper layers need to be communicated back and forth with the server. For instance, several prior works study how to fine-tune Vision Transformers in FL [30, 53, 54, 55, 56], how to adapt Contrastive Language Image Pre-training (CLIP [57]) to FL [58], and how to fine-tune using Neural-Tangent Kernels [59].

Ridge Regression in Distributed and Federated Learning. Regarding prior works on ridge regression in FL settings, [60] finds an optimal Tikhonov Regularized Least Squares solution for a federation of only two clients that in practice constitutes a cross-knowledge distillation framework, and [61] and [62] apply RR to a Vertical FL scenario in which the feature space varies among the clients but the sample space is the same. To the best of our knowledge, no prior works employ ridge regression in the Horizontal cross-device FL scenario, where the feature space is shared among the clients but the sample space varies. This is the first work to show the effectiveness and efficiency of RR in real-world, horizontal, cross-device FL problems.

Other work generalizes Ridge Regression to the Distributed Learning setting [63, 64]. While similar in spirit, Distributed Learning is fundamentally different from FL as privacy is not a concern and data is i.i.d. among the clients.

B The rebalancing matrix Γ

In this work, we normalize the diagonal of the rebalancing matrix Γ, originally introduced by [33], such that the sum of its diagonal entries is n. This choice is guided by the intuition that, without regularization on Γ, in the scenario of uniform class frequencies, the regularization matrix Γ would have $1/C$ as diagonal entries for each class. Consequently, the Γ matrix acts as a rescaling factor of $(1/C)^\alpha$ for the final optimal weights W^*.

To compute the RR classifier with class rebalancing, we need the empirical frequency of each class. Therefore, in FED3R (see Algorithm 1) each client has to communicate one additional vector $g_k = Y_k^\top \mathbf{1},\ \mathbf{1} \in \mathbb{R}^{n_k}$, with the server. Together with A and b, the server aggregates all the g_k to obtain the final vector $g = \sum_{k \in \mathcal{K}} g_k$, compute $\Gamma = diag\left(C \frac{\|g\|_1}{g}\right)$ and, finally, solves Eq. 2.

C Details on the FED3R-FT algorithm

The pseudo-code for the FED3R-FT algorithm is outlined in Algorithm 2. FED3R-FT can be viewed as a more general version of FED3R, with three minor distinctions that we summarize below:

1. The UpdateRule function, nested in the ClientUpdate function, is redundant in FED3R since FED3R does not need to update the classifier parameters ϕ.
2. The sampling strategy could be significantly simplified and optimized. Unlike FED3R-FT, FED3R does not necessitate a strict division into rounds because the clients only need to be sampled once.
3. In FED3R, both the clients and the server do not require internal states. In other words, it suffices to *1)* set $\tilde{A}_k^{\text{old}} = \mathbf{0}$ and $\tilde{b}_k^{\text{old}} = \mathbf{0}$, *2)* communicate the g_k statistics only once concurrently with the sole occasion each client k is sampled, and that *c)* the server then computes A and b only once all the clients have transmitted their statistics.

Algorithm 2: FED3R-FT

Require:
Server $\mathcal{S}$, clients $k \in \mathcal{K}$, local datasets $\mathcal{D}_k$
Model $\mathcal{M} : \mathcal{X} \to \mathbb{R}^C$, feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$, classifier $\phi : \mathbb{R}^d \to \mathbb{R}^C : \mathcal{M}\left(x; \theta^0\right) = \phi\left(\varphi\left(x; \theta_\varphi^0\right); \theta_\phi^0\right)$
Number of rounds T, Number of local epochs E, Number of clients sampled per each round $K \leq |\mathcal{K}|$
Hyper-parameters $\lambda > 0, \alpha > 0$
Initialize:
$A^0 = \lambda I_d \in \mathbb{R}^{d \times d}$, $b^0 = \mathbf{0} \in \mathbb{R}^{d \times C}$, $g = \mathbf{0} \in \mathbb{R}^C$,
$\tilde{A}_k^{\text{old}} = \mathbf{0} \in \mathbb{R}^{d \times d}\ \forall k \in \mathcal{K}$, $\tilde{b}_k^{\text{old}} = \mathbf{0} \in \mathbb{R}^{d \times C}\ \forall k \in \mathcal{K}$, $s = \mathbf{0} \in \mathbb{R}^C$
Server
  for *each* *round* $t \in [T]$ **do**
    Randomly extract K clients $\mathcal{K}' \subset \mathcal{K}$
    for $k \in \mathcal{K}'$ *in parallel* **do**
      $\theta_k^t, \tilde{A}_k^t, \tilde{b}_k^t, g_k$ = **ClientUpdate**$\left(\theta^{t-1}\right)$
    $\theta^t = \text{Aggregate}\left(\theta_1^t, \theta_2^t, \ldots, \theta_K^t\right)$
    $\tilde{A}^t = \sum_k \tilde{A}_k^t$, $\tilde{b}^t = \sum_k \tilde{b}_k^t$, $g \leftarrow g + \sum_k g_k$ `// Aggregate the stats received from the clients`
    $A^t = A^{t-1} + \tilde{A}^t$, $b^t = b^{t-1} + \tilde{b}^t$ `// Update` A `and` b
    Only when needed for inference: Γ^t = **GammaUpdate**(C, g), $W^t = (A^t)^{-1} b^t \left(\Gamma^t\right)^\alpha$
  Return W^t
ClientUpdate
  Extract Z_k
  `// Get the delta statistics`
  $\tilde{A}_k^{\text{new}} = (Z_k^t)^\top Z_k^t$ $\tilde{A}_k^t = \tilde{A}_k^{\text{new}} - \tilde{A}_k^{\text{old}}$ $\tilde{A}_k^{\text{old}} = \tilde{A}_k^{\text{new}}$
  $\tilde{b}_k^{\text{new}} = (Z_k^t)^\top Y_k$ $\tilde{b}_k^t = \tilde{b}_k^{\text{new}} - \tilde{b}_k^{\text{old}}$ $\tilde{b}_k^{\text{old}} = \tilde{b}_k^{\text{new}}$
  $g_k = (1 - s_k) Y_k^\top \mathbf{1}$, $\mathbf{1} \in \mathbb{R}^{n_k}$ $s_k = 1$ `//Send the img/class stats` $\iff$ `first time, else send` $\mathbf{0}$
  $\theta_k^t = \text{UpdateRule}\left(\mathcal{M}, \theta^{t-1}, X_k, Y_k\right)$
  Return $\theta_k^t, \tilde{A}_k^t, \tilde{b}_k^t, g_k$
GammaUpdate
  $f := C \frac{\|g\|_1}{g}$ `// Some classes might not be seen yet. Thus, some` ∞ `values may appear`
  Substitute eventual ∞ values in f with the max value of f among the values $< \infty$
  Return $\Gamma^t = \text{diag}\,(f)$

D Privacy of FED3R

In the context of the FED3R algorithm, clients have to transmit the A_k and b_k statistics, along with the vector g_k that reflects the distribution of their labels. Some may express concerns about the potential information leakage inherent in sharing this data, which extends beyond the disclosure associated with merely sharing model weights or gradients. However, it is crucial to note that any information the clients send to the server only needs to be aggregated. In other words, the server does not necessitate accessing individual values but rather needs solely to use the aggregated results. Therefore, privacy can be easily addressed by employing the Secure Aggregation protocol [65].

E Implementation details

As explained in Section 3, FED3R with fixed feature extractor requires a single communication with each client, which can happen once they are ready. However, to ensure a fair comparison with gradient-based FL algorithms which are sensitive to different client samplings, we orchestrate the

Table E.1: Communication costs and latent feature space dimensionality of the architectures adopted in this work.

Architecture	Classifier	Feature Extractor	Model	Feature space Dimensionality
MobileNetV2 [43]	12.00 MB	26.60 MB	38.6 MB	1280
ViT [45]	9.68 MB	1.22 GB	1.22 GB	1024
SWIN [46]	14.40 MB	901.44 MB	915.84 MB	1536
ConvNext [47]	14.40 MB	904.80 MB	919.20 MB	1536

training in rounds as in a standard FL setting in all the experiments, even the ones with the fixed feature extractor. However, in this case, the total FED3R communication costs are overestimated because we account for unnecessary communication when the same clients are sampled again due to clients sharing the same statistics. Despite this, the communication requirements are typically less than or comparable to the other methods.

We run all the experiments using an NVIDIA A100-SXM4-40GB on the Landmarks-Users-160K [44] dataset using the FL clients partition provided by [3]. The dataset has a total of 2028 classes and 164172 images distributed among the 1262 clients. When not differently stated, we use a MobileNetV2 [43] network. We run all the experiments for 3000 rounds, always sampling 10 clients per round except when differently declared. When we need to train the whole model or the classifier only, we use SGD as the optimizer, with a learning rate of 0.1 and a weight decay of $4 \cdot 10^{-5}$, a batch size of 50 and 5 local epochs. When we need to aggregate the model parameters' updates of the clients at the end of the round, we use the classic SGD server optimizer [66] with a learning rate equal to 1.0 and no momentum. Regarding the FED3R hyper-parameters, we always set $\lambda = 0.01$ and $\alpha = 0.8$ because we found that these values consistently provide the best results.

Moreover, we compare our methods with two gradient-based algorithms from the literature, namely FedAvg [1] and Scaffold [16]. We did not include Mime and MimeLite [15] because they failed to converge in the FED3R-FT setting. Furthermore, we compare our results with a modified version of the only partially non-gradient-based FL method found in the literature, FedNCM [28]. In contrast to FED3R, FedNCM involves two phases. The first phase constructs a classification head using Nearest Class Means, while the second phase fine-tunes the entire model with a gradient-based algorithm. In our implementation, we exclude the second stage to be fair with our single-stage algorithm. Additionally, we allow the use of FedNCM even when the feature extractor is not fixed, adopting the same strategy as FED3R-FT for the feature maps. Specifically, we consider only the most recent ones, even if they belong to different latent spaces due to the feature extractor updates.

Furthermore, as in [28], we conduct a study on the communication costs required by the different methods to determine the applicability of the algorithms in real-world situations. Table E.1 illustrates, for all the adopted feature extractors, the basic costs (in bytes) of the complete models, feature extractors, and classifiers, which need to be communicated multiple times and in various modes concerning different methods. Moreover, the table shows also the latent space's embedding dimension.

In the forthcoming sections, we present a comparative analysis involving FED3R-FT, FED3R-RF with 5000 random features (FED3R-RF D=5k), and FED3R-RF with 10000 random features (FED3R-RF D=10k), for all the experiments.

F Communication costs computation

The costs presented in Tables 1, G.1, and G.2 have been derived from the initial values in E.1. In all experiments, like in [28], we assume that the model's parameters precision is FP32. The columns labeled *Downstr./k* and *Upstream/k* represent the aggregate downstream and upstream communication requirements per client per round, respectively. These values depend on the specific characteristics of the algorithm under consideration. Likewise, the columns labeled *Downstream* and *Upstream* reflect the overall downstream and upstream communication needs for all clients sampled in a given round. Consequently, the *Round/k* and *Round* columns denote the respective sums of downstream and upstream costs. Let m, b and c be the size of the whole model, the feature extractor, and the classifier, respectively. From these provided definitions, it is evident that $m = b + c$.

Below we briefly summarize how the *Downstr./k* and *Upstream/k* costs have been calculated per each algorithm, and eventual additional costs:

Table F.1: Pre-training strategies, total prior costs when the feature extractor is fixed and total costs for FED3R algorithm with optimal sampling strategy of 1262 clients of the Landmarks-Users-160K dataset [3]. All pre-trained models except MobileNetV2 are sourced from [67].

Architecture	Pre-training strategy	Feature Extractor size	Prior costs	Algorithm	Round/k	Total cost
MobileNetV2 [43]	ImageNet-1k	26.6 MB	33.57 GB	FED3R	16.95 MB	21.39 GB
				FED3R-RF $D = 5k$	140.57 MB	177.40 GB
				FED3R-RF $D = 10k$	481.13 MB	607.19 GB
ViT [45]	Pre-trained on LAION-2B using CLIP [57], fine-tuned on ImageNet-12k and ImageNet-1k	1.22 GB	1.54 TB	FED3R	12.51 MB	15.79 GB
				FED3R-RF $D = 5k$	140.57 MB	177.40 GB
				FED3R-RF $D = 10k$	481.13 MB	607.19 GB
SWIN [46]	ImageNet-22k [68]	901.44 MB	1.14 TB	FED3R	21.91 MB	27.65 GB
				FED3R-RF $D = 5k$	140.57 MB	177.40 GB
				FED3R-RF $D = 10k$	481.13 MB	607.19 GB
ConvNext [47]	ImageNet-22k [68]	919.20 MB	1.16 TB	FED3R	21.91 MB	27.65 GB
				FED3R-RF $D = 5k$	140.57 MB	177.40 GB
				FED3R-RF $D = 10k$	481.13 MB	607.19 GB

- **FedAvg [1].** Each sampled client downloads and uploads the model only once: *Downstr./k* = m, *Upstream/k* = m.
- **Scaffold [16].** Each sampled client downloads and uploads both the model and its control variate: *Downstr./k* = $2m$, *Upstream/k* = $2m$.
- **FED3R, FED3R-RF (fixed feature extractor).** Each client needs to receive the feature extractor parameters only once. If we do not assume clients already have the feature extractor parameters before the training begins (though this assumption is reasonable in scenarios where the server, as a business, deploys its application and may have already incorporated these parameters in the clients' software), there is an additional communication cost of bK. Except that for these costs, each sampled client does not need to download any information from the server, but it needs to upload the local statistics A_k, b_k, g_k to the server: *Downstr./k* = 0, *Upstream/k* = $d^2 + dC + C$. If we are using FED3R-RF the upstream costs per client are *Upstream/k* = $D^2 + DC + C$ instead.
- **FED3R-FT, FED3R-RF (fine-tuned feature extractor)**. In this case, it is necessary to combine both the downstream and upstream costs with the corresponding costs of the gradient-based FL algorithm used to update the model parameters. The sole FED3R-FT or FED3R-RF contribution is equivalent to the previous case.
- **FedNCM [28].** As already specified in Section E, in our experiments we considered only the first stage of the algorithm, without the phase of the fine-tuning of the feature extractor. If the feature extractor is fixed, there are no downstream costs, but each sampled client still needs to communicate the class centroids upstream: *Downstr./k* = 0, *Upstream/k* = dC. Likewise FED3R, the eventual prior costs are bK. In addition, if the feature extractor is not fixed, we have to sum the costs of the underlying gradient-based algorithm.

The total communication costs to share the feature extractor parameters with all the clients, and the maximum optimal costs for all the architectures to sample all the clients once using FED3R, are shown in Table F.1.

G Additional details on the experiments presented in the main paper

In this section, we expound upon additional considerations and experiments that supplement those presented in the main paper.

G.1 Experiments with fine-tuned feature extractor

This section provides further details regarding the experiments with the fine-tuned feature extractor presented in Section 4.

Table G.1 shows the communication costs and the accuracy achieved at round 3000. It is interesting to note that the total costs to converge to 50% accuracy are just slightly higher in this case than with the fixed feature extractor (see Table 1), but fewer rounds are required.

Moreover, Figure G.1 displays how, for the same experiments, the gap between FedAvg and Scaffold is much higher for the methods that employ the softmax classifier, while it is relatively small for the FED3R methods, meaning that Scaffold improves the classifier more than the feature extractor. Therefore, FED3R can also be used as a tool to discern the quality of the feature maps.

Table G.1: Communication costs and final performance for the experiments with fine-tuned feature extractor.

Algorithm	Downstr./k	Upstream/k	Round/k	Downstream	Upstream	Round	30% acc	40% acc	50% acc	60% acc	Acc r=3k (%)
FedAvg	38.60 MB	38.60 MB	77.20 MB	386.00 MB	386.00 MB	772.00 MB	228.51 GB	386.00 GB	771.23 GB	-	58.17
Scaffold	77.20 MB	77.20 MB	154.40 MB	772.00 MB	772.00 MB	1.54 GB	273.29 GB	433.86 GB	747.30 GB	**1.60 TB**	62.32
FED3R-FT FedAvg	38.60 MB	55.55 MB	94.15 MB	386.00 MB	555.45 MB	941.45 MB	**67.78 GB**	**161.93 GB**	728.68 GB	-	52.95
FED3R-FT Scaffold	77.20 MB	94.15 MB	171.35 MB	772.00 MB	941.45 MB	1.71 GB	116.51 GB	287.86 GB	937.26 GB	-	54.02
FED3R-RF 5k FedAvg	38.60 MB	179.17 MB	217.77 MB	386.00 MB	1.79 GB	2.18 GB	113.24 GB	206.88 GB	**699.04 GB**	-	58.6
FED3R-RF 5k Scaffold	77.20 MB	217.77 MB	294.97 MB	772.00 MB	2.18 GB	2.95 GB	153.38 GB	289.07 GB	755.12 GB	3.18 TB	60.11
FED3R-RF 10k FedAvg	38.60 MB	519.73 MB	558.33 MB	386.00 MB	5.20 GB	5.58 GB	262.41 GB	480.16 GB	1.18 TB	6.01 TB	61.61
FED3R-RF 10k Scaffold	77.20 MB	558.33 MB	635.53 MB	772.00 MB	5.58 GB	6.36 GB	305.05 GB	559.26 GB	1.13 TB	4.19 TB	**62.69**
FedNCM FedAvg	38.60 MB	48.98 MB	87.58 MB	386.00 MB	489.83 MB	875.83 MB	131.38 GB	367.85 GB	1.78 TB	-	50.7
FedNCM Scaffold	77.20 MB	87.58 MB	164.78 MB	772.00 MB	875.83 MB	1.65 GB	229.05 GB	430.08 GB	1.35 TB	-	53.49

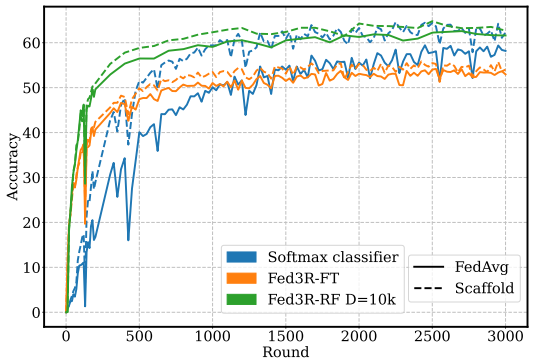


Figure G.1: FED3R-FT comparison between FedAvg and Scaffold.

G.2 Experiments with more complex architectures

This section provides additional details regarding the experiments in the *Enabling Foundation Models in FL* paragraph of Section 4.

Figure G.2 visually emphasizes the notable speed advantage of the FED3R methods compared to other approaches. Furthermore, Table G.2 provides insight into the associated communication costs. It is noteworthy that the FED3R methods provide always the best performance while needing lower communication expenses up to the target of 70% accuracy. Additionally, utilizing feature extractors with enhanced generalization capabilities enables savings in total communication to achieve the target accuracy compared to the use of lightweight networks like MobileNetV2 [43].

H Centralized RR results using the random features

The outcomes of centralized experiments employing random features to approximate the RBF kernel empirically show that augmenting the number of random features significantly enhances performance. Specifically, Figure H.1 illustrates how, with the random features approximation, the performance of RR calculated over the feature maps provided by the feature extractor across the entire dataset eventually approaches the upper bound established by the exact KRR solution on a subset of the dataset, where a maximum of 40 images per class is considered. It is noteworthy that the exact KRR solution was not computed over the entire dataset due to computational constraints. Indeed, the exact solution would require storing a kernel matrix of dimensionality $n \times n$, where $n = 164172$ for Landmarks-Users-160K. Nevertheless, utilizing the whole dataset or increasing the number of random features should theoretically improve results further. In addition, Figure H.1b empirically shows that KRR can even yield superior performance compared to a softmax classifier at convergence.

I Analysis on the number of clients sampled per round

In this section, we investigate the impact of varying the number of clients $\kappa = |\mathcal{K}'|$ sampled per round with uniform probability on FED3R-FT. We reintroduce the concept of rounds even for FED3R for a fair comparison with round-scheduled methods as in all the other FED3R experiments presented in this paper, despite more advanced sampling strategies to reduce costs further would be preferable (see Appendix F).

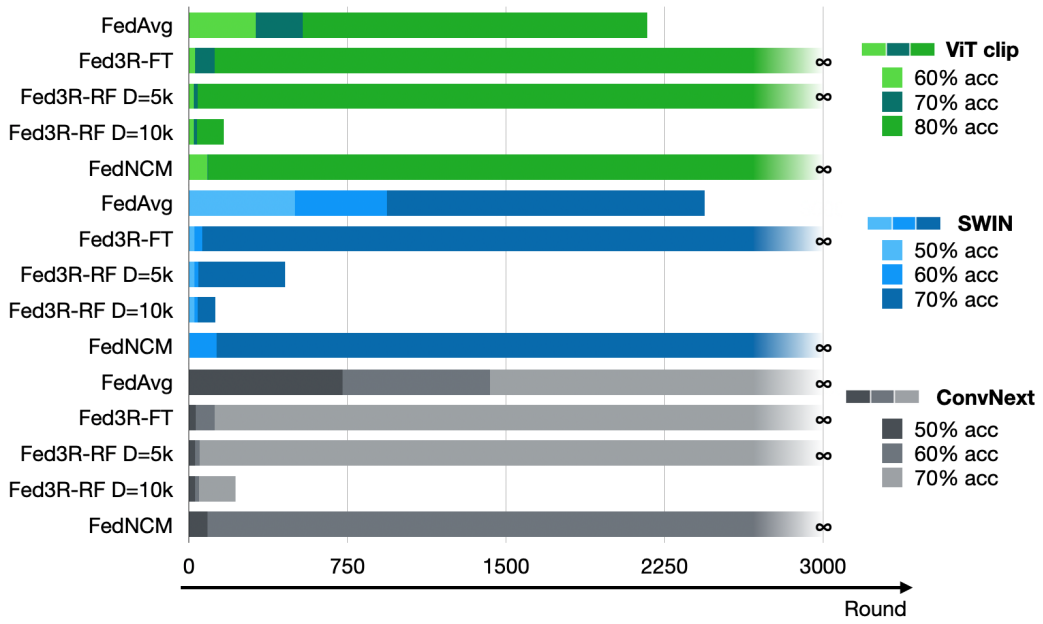


Figure G.2: Number of rounds required to reach the target accuracy for the complex feature extractors and the methods of the experiments presented in the *Enabling Foundation Models in FL* paragraph of Section 4.

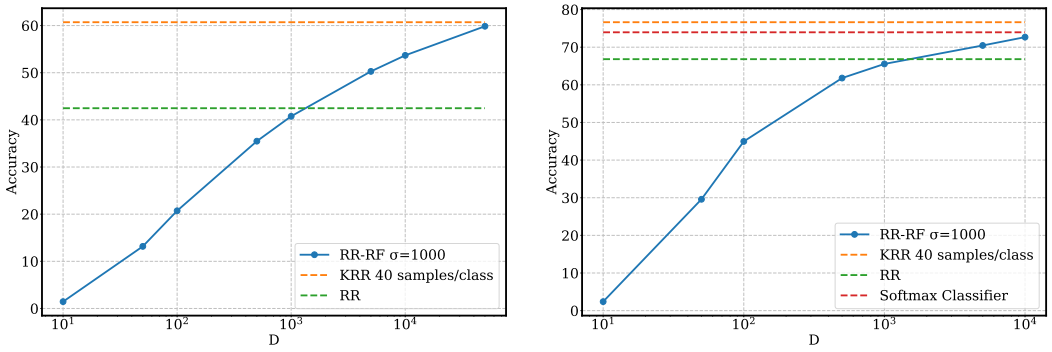
Table G.2: Communication costs and final performance for the experiments with the complex feature extractors.

Architecture	Algorithm	Downstr./k	Upstream/k	Round/k	Downstream	Upstream	Round	30% acc	40% acc	50% acc	60% acc	70% acc	80% acc	Acc r=3k (%)
ViT [45]	FedAvg	9.60 MB	9.60 MB	19.20 MB	96.00 MB	96.00 MB	192.00 MB	18.05 GB	26.69 GB	40.90 GB	60.29 GB	103.49 GB	**416.26 GB**	81.23
	FED3R	0.00 B	16.95 MB	16.95 MB	0.00 B	169.45 MB	169.45 MB	**1.19 GB**	**1.69 GB**	**2.71 GB**	**4.91 GB**	20.67 GB	-	72.11
	FED3R-RF $D = 5k$	0.00 B	140.57 MB	140.57 MB	0.00 B	1.41 GB	1.41 GB	9.84 GB	12.65 GB	19.68 GB	30.92 GB	57.63 GB	-	79.45
	FED3R-RF $D = 10k$	0.00 B	481.13 MB	481.13 MB	0.00 B	4.81 GB	4.81 GB	52.92 GB	72.17 GB	86.60 GB	110.66 GB	187.64 GB	793.86 GB	**81.49**
	FedNCM	0.00 B	8.31 MB	8.31 MB	0.00 B	83.07 MB	83.07 MB	5.48 GB	5.73 GB	6.81 GB	7.14 GB	**7.48 GB**	-	73.22
SWIN [46]	FedAvg	14.40 MB	14.40 MB	28.80 MB	144.00 MB	144.00 MB	288.00 MB	58.18 GB	89.28 GB	144.58 GB	269.86 GB	702.43 GB	-	71.03
	FED3R	0.00 B	21.91 MB	21.91 MB	0.00 B	219.05 MB	219.05 MB	**1.97 GB**	**3.07 GB**	**5.48 GB**	**13.36 GB**	-	-	65.35
	FED3R-RF $D = 5k$	0.00 B	140.57 MB	140.57 MB	0.00 B	1.41 GB	1.41 GB	12.65 GB	19.68 GB	33.74 GB	61.85 GB	636.77 GB	-	70.35
	FED3R-RF $D = 10k$	0.00 B	481.13 MB	481.13 MB	0.00 B	4.81 GB	4.81 GB	72.17 GB	86.60 GB	115.47 GB	202.07 GB	**596.60 GB**	-	**73.17**
	FedNCM	0.00 B	12.46 MB	12.46 MB	0.00 B	124.60 MB	124.60 MB	8.47 GB	10.22 GB	10.84 GB	26.29 GB	-	-	61.23
ConvNext [47]	FedAvg	14.40 MB	14.40 MB	28.80 MB	144.00 MB	144.00 MB	288.00 MB	69.98 GB	117.50 GB	209.09 GB	410.11 GB	-	-	66.92
	FED3R	0.00 B	21.91 MB	21.91 MB	0.00 B	219.05 MB	219.05 MB	**1.97 GB**	**3.29 GB**	**6.57 GB**	**26.29 GB**	-	-	62.69
	FED3R-RF $D = 5k$	0.00 B	140.57 MB	140.57 MB	0.00 B	1.41 GB	1.41 GB	14.06 GB	22.49 GB	37.95 GB	70.28 GB	-	-	68.43
	FED3R-RF $D = 10k$	0.00 B	481.13 MB	481.13 MB	0.00 B	4.81 GB	4.81 GB	72.17 GB	91.41 GB	129.90 GB	221.32 GB	**1.05 TB**	-	**71.43**
	FedNCM	0.00 B	12.46 MB	12.46 MB	0.00 B	124.60 MB	124.60 MB	8.60 GB	10.34 GB	10.96 GB	-	-	-	58.47

Table I.1 details the communication costs when the feature extractor is non-fixed. Notably, to minimize the overall communication required to achieve the target accuracy, reducing the number of sampled clients κ is more effective. However, it is worth observing that performance improves with more clients sampled per round.

Table I.2 presents the communication costs when the feature extractor is fixed. In this scenario, variations in communication costs depend solely on the order in which clients are sampled. If the sampling order remains consistent, we would expect that the total communication required to reach the target performance remains the same, as the server only needs to collect statistics once from each client. This is further evident when examining Figure I.1: while the FED3R-FT and FED3R-RF curves are noisy when fine-tuning the feature extractor (Figure I.1a), they are much more stable and exhibit performance proportionate to κ when the feature extractor is fixed (I.1b). Moreover, all the curves converge to the same RR upper bound (or RR-RF upper bound for FED3R-RF), regardless of the value of κ.

FED3R ensures convergence once each client has been sampled at least once. The expected number of rounds needed to achieve this, employing uniform probability and sampling κ clients per round, aligns with the Batch Coupon Collector's Problem [69, 70, 71]. Figure I.2 illustrates the results of our simulations for the Batch Coupon Collector's Problem while varying κ with a total of $K = 1262$ clients. Comparing the outcomes in Figure I.1b and Figure I.2, it is evident that, in practice, significantly fewer communication rounds are required to reach convergence. This discrepancy arises because sampling new unseen clients becomes progressively more challenging as more distinct clients



(a) Pre-trained on ImageNet

(b) Fine-tuned on Landmarks-Users-160K

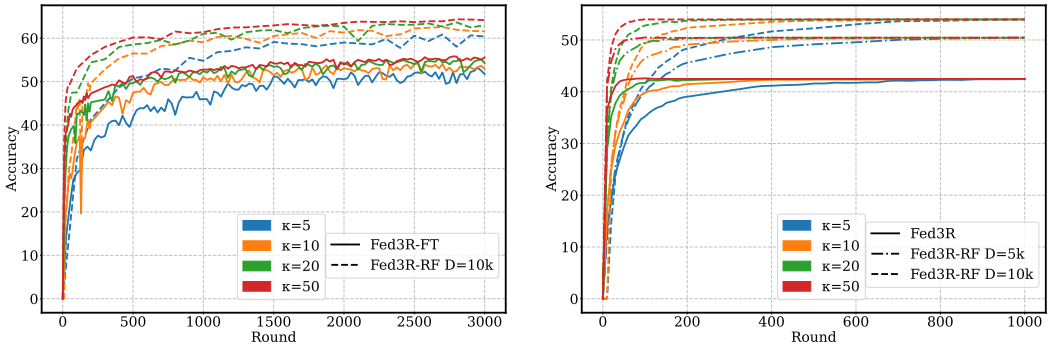
Figure H.1: (centralized) RR using D random features to approximate the RBF kernel compared to the exact KRR solution with RBF kernel computed over a subset of the whole Landmarks-Users-160K dataset where there are at most 40 images per class, using the MobileNetV2 [43] architecture. We keep $\sigma = 1000$ for both KRR and RR-RF.

Table I.1: Comparative analysis of communication costs by varying the number of clients κ per round the non-fixed feature extractor.

Algorithm	κ	**Downstr./k**	**Upstream/k**	**Round/k**	**Downstream**	**Upstream**	**Round**	**30% acc**	**40% acc**	**50% acc**	**60% acc**	**Acc r=3k (%)**
FedAvg	5	38.60 MB	38.60 MB	77.20 MB	193.00 MB	193.00 MB	386.00 MB	162.12 GB	325.01 GB	**549.28 GB**	-	55.29
FED3R-FT	5	38.60 MB	55.55 MB	94.15 MB	193.00 MB	277.73 MB	470.73 MB	59.78 GB	**152.04 GB**	698.56 GB	-	51.75
FED3R-RF $D = 5k$	5	38.60 MB	179.17 MB	217.77 MB	193.00 MB	895.84 MB	1.09 GB	105.62 GB	260.23 GB	908.09 GB	-	57.83
FED3R-RF $D = 10k$	5	38.60 MB	519.73 MB	558.33 MB	193.00 MB	2.60 GB	2.79 GB	262.41 GB	519.25 GB	1.30 TB	6.92 TB	60.4
FedNCM	5	38.60 MB	48.98 MB	87.58 MB	193.00 MB	244.92 MB	437.92 MB	118.68 GB	358.22 GB	1.16 TB	-	50.55
FedAvg	10	38.60 MB	38.60 MB	77.20 MB	386.00 MB	386.00 MB	772.00 MB	228.51 GB	386.00 GB	771.23 GB	-	58.17
FED3R-FT	10	38.60 MB	55.55 MB	94.15 MB	386.00 MB	555.45 MB	941.45 MB	67.78 GB	161.93 GB	728.68 GB	-	52.95
FED3R-RF $D = 5k$	10	38.60 MB	179.17 MB	217.77 MB	386.00 MB	1.79 GB	2.18 GB	113.24 GB	206.88 GB	699.04 GB	-	58.6
FED3R-RF $D = 10k$	10	38.60 MB	519.73 MB	558.33 MB	386.00 MB	5.20 GB	5.58 GB	262.41 GB	480.16 GB	1.18 TB	6.01 TB	61.61
FedNCM	10	38.60 MB	48.98 MB	87.58 MB	386.00 MB	489.83 MB	875.83 MB	131.38 GB	367.85 GB	1.78 TB	-	50.7
FedAvg	20	38.60 MB	38.60 MB	77.20 MB	772.00 MB	772.00 MB	1.54 GB	416.88 GB	628.41 GB	1.29 TB	**3.57 TB**	59.73
FED3R-FT	20	38.60 MB	55.55 MB	94.15 MB	772.00 MB	1.11 GB	1.88 GB	**41.42 GB**	182.64 GB	975.34 GB	-	54.28
FED3R-RF $D = 5k$	20	38.60 MB	179.17 MB	217.77 MB	772.00 MB	3.58 GB	4.36 GB	82.75 GB	161.15 GB	779.61 GB	9.76 TB	60.16
FED3R-RF $D = 10k$	20	38.60 MB	519.73 MB	558.33 MB	772.00 MB	10.39 GB	11.17 GB	201.00 GB	335.00 GB	1.53 TB	10.43 TB	62.99
FedNCM	20	38.60 MB	48.98 MB	87.58 MB	772.00 MB	979.67 MB	1.75 GB	236.48 GB	555.28 GB	2.44 TB	-	51.85
FedAvg	50	38.60 MB	38.60 MB	77.20 MB	1.93 GB	1.93 GB	3.86 GB	799.02 GB	1.44 TB	2.79 TB	7.80 TB	62.43
FED3R-FT	50	38.60 MB	55.55 MB	94.15 MB	1.93 GB	2.78 GB	4.71 GB	47.07 GB	178.88 GB	1.85 TB	-	55.57
FED3R-RF $D = 5k$	50	38.60 MB	179.17 MB	217.77 MB	1.93 GB	8.96 GB	10.89 GB	98.00 GB	185.10 GB	990.84 GB	16.90 TB	60.95
FED3R-RF $D = 10k$	50	38.60 MB	519.73 MB	558.33 MB	1.93 GB	25.99 GB	27.92 GB	251.25 GB	390.83 GB	1.62 TB	13.51 TB	**64.16**
FedNCM	50	38.60 MB	48.98 MB	87.58 MB	1.93 GB	2.45 GB	4.38 GB	218.96 GB	1.06 TB	5.84 TB	-	52.74

Table I.2: Comparative analysis of communication costs by varying the number of clients κ per round with fixed feature extractor.

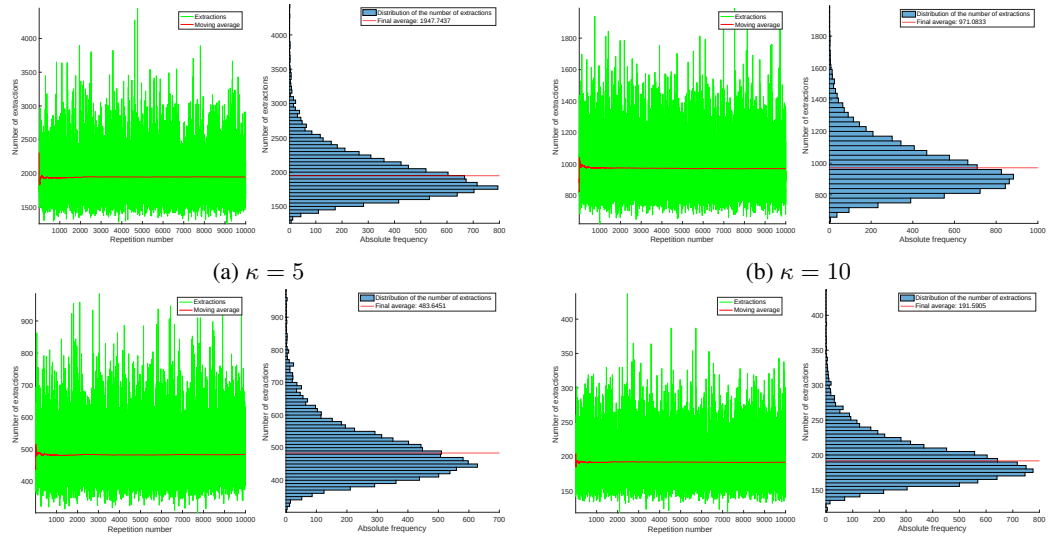
Algorithm	κ	**Downstr./k**	**Upstream/k**	**Round/k**	**Downstream**	**Upstream**	**Round**	**30% acc**	**40% acc**	**50% acc**	**Acc r=3k (%)**
FedAvg	5	12.00 MB	12.00 MB	24.00 MB	60.00 MB	60.00 MB	120.00 MB	149.76 GB	312.00 GB	-	40.96
FED3R	5	0.00 B	16.95 MB	16.95 MB	0.00 B	84.73 MB	84.73 MB	4.58 GB	23.81 GB	-	42.48
FED3R-RF $D = 5k$	5	0.00 B	140.57 MB	140.57 MB	0.00 B	702.84 MB	702.84 MB	30.92 GB	70.99 GB	487.07 GB	50.41
FED3R-RF $D = 10k$	5	0.00 B	481.13 MB	481.13 MB	0.00 B	2.41 GB	2.41 GB	108.25 GB	211.70 GB	683.20 GB	**53.97**
FedNCM	5	0.00 B	10.38 MB	10.38 MB	0.00 B	51.92 MB	51.92 MB	6.75 GB	-	-	35.94
FedAvg	10	12.00 MB	12.00 MB	24.00 MB	120.00 MB	120.00 MB	240.00 MB	237.36 GB	542.40 GB	-	40.79
FED3R	10	0.00 B	16.95 MB	16.95 MB	0.00 B	169.45 MB	169.45 MB	5.42 GB	17.96 GB	-	42.47
FED3R-RF $D = 5k$	10	0.00 B	140.57 MB	140.57 MB	0.00 B	1.41 GB	1.41 GB	37.95 GB	78.72 GB	560.87 GB	50.4
FED3R-RF $D = 10k$	10	0.00 B	481.13 MB	481.13 MB	0.00 B	4.81 GB	4.81 GB	129.90 GB	250.19 GB	553.30 GB	**53.96**
FedNCM	10	0.00 B	10.38 MB	10.38 MB	0.00 B	103.83 MB	103.83 MB	9.35 GB	-	-	35.93
FedAvg	20	12.00 MB	12.00 MB	24.00 MB	240.00 MB	240.00 MB	480.00 MB	402.24 GB	909.12 GB	-	43.16
FED3R	20	0.00 B	16.95 MB	16.95 MB	0.00 B	338.90 MB	338.90 MB	**4.07 GB**	**17.28 GB**	-	42.48
FED3R-RF $D = 5k$	20	0.00 B	140.57 MB	140.57 MB	0.00 B	2.81 GB	2.81 GB	28.11 GB	59.04 GB	**390.78 GB**	50.44
FED3R-RF $D = 10k$	20	0.00 B	481.13 MB	481.13 MB	0.00 B	9.62 GB	9.62 GB	96.23 GB	192.45 GB	500.37 GB	**53.98**
FedNCM	20	0.00 B	10.38 MB	10.38 MB	0.00 B	207.67 MB	207.67 MB	8.31 GB	-	-	35.94
FedAvg	50	12.00 MB	12.00 MB	24.00 MB	600.00 MB	600.00 MB	1.20 GB	950.40 GB	2.23 TB	-	43.94
FED3R	50	0.00 B	16.95 MB	16.95 MB	0.00 B	847.25 MB	847.25 MB	7.63 GB	20.33 GB	-	42.48
FED3R-RF $D = 5k$	50	0.00 B	140.57 MB	140.57 MB	0.00 B	7.03 GB	7.03 GB	56.23 GB	70.28 GB	400.62 GB	50.41
FED3R-RF $D = 10k$	50	0.00 B	481.13 MB	481.13 MB	0.00 B	24.06 GB	24.06 GB	192.45 GB	240.56 GB	625.47 GB	**53.95**
FedNCM	50	0.00 B	10.38 MB	10.38 MB	0.00 B	519.17 MB	519.17 MB	9.86 GB	-	-	35.94



(a) Fine-tuned feature extractor

(b) Fixed feature extractor

Figure I.1: FED3R-FT comparison varying the number clients sampled per round κ.



(a) $\kappa = 5$

(b) $\kappa = 10$

(c) $\kappa = 20$

(d) $\kappa = 50$

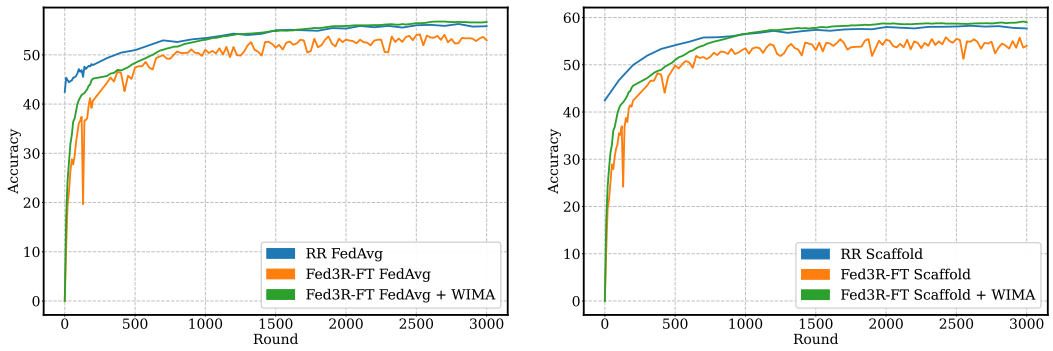
Figure I.2: Simulations of the expected number of clients we need to sample to extract each client at least once, for the Landmarks-Users-160K federated datasets with 1262 clients.

have already been sampled. Therefore, we can assume that most of the clients have been sampled earlier.

J FED3R-FT converges to the RR upper bound using WIMA

As highlighted in Section 3.2, FED3R-FT permits fine-tuning the feature extractor to enhance latent space separability, albeit at the expense of perturbing the optimal solution achievable through FED3R. To address this issue and stabilize the latent space, we leverage Window-based Weight Averaging (WIMA) [72]. WIMA, initially designed for standard gradient-based optimization algorithms, offers a straightforward yet effective approach to balance the training process and reduce convergence time. As depicted in Figure J.1, using WIMA allows FED3R-FT to reach the RR upper-bound, simultaneously enhancing overall performance. We opted for a moving window size of 370 rounds based on the results of Figure I.1b, which shows how this number of rounds is sufficient for approaching convergence with $\kappa = 10$. This suggests that, within a window of this size, a substantial portion of the clients has likely been included in the sampling process.

Finally, Table J.1 provides the communication costs for the FED3R-FT experiments with WIMA. Similarly to Table G.1, only Scaffold can provide comparable results with FED3R-FT (or FED3R-RF), despite being much slower.



(a) Feature extractor fine-tuned with FedAvg

(b) Feature extractor fine-tuned with Scaffold

Figure J.1: FED3R-FT using WIMA [72].

Table J.1: Communication costs for the WIMA experiments.

Algorithm	Downstr./k	Upstream/k	Round/k	Downstream	Upstream	Round	30% acc	40% acc	50% acc	60% acc	Acc r=3k (%)
FedAvg	38.60 MB	38.60 MB	77.20 MB	386.00 MB	386.00 MB	772.00 MB	328.10 GB	431.55 GB	646.94 GB	1.43 TB	63.05
Scaffold	77.20 MB	77.20 MB	154.40 MB	772.00 MB	772.00 MB	1.54 GB	443.13 GB	631.50 GB	822.95 GB	**1.26 TB**	**68.30**
FED3R-FT FedAvg	38.60 MB	55.55 MB	94.15 MB	386.00 MB	555.45 MB	941.45 MB	**32.95 GB**	**84.73 GB**	588.41 GB	-	56.69
FED3R-FT Scaffold	77.20 MB	94.15 MB	171.35 MB	772.00 MB	941.45 MB	1.71 GB	61.68 GB	159.35 GB	772.77 GB	-	59.01
FED3R-FT $D = 5k$ FedAvg	38.60 MB	179.17 MB	217.77 MB	386.00 MB	1.79 GB	2.18 GB	65.33 GB	130.66 GB	**389.80 GB**	2.77 TB	62.56
FED3R-FT $D = 5k$ Scaffold	77.20 MB	217.77 MB	294.97 MB	772.00 MB	2.18 GB	2.95 GB	91.44 GB	185.83 GB	492.60 GB	1.96 TB	64.98
FED3R-FT $D = 10k$ FedAvg	38.60 MB	519.73 MB	558.33 MB	386.00 MB	5.20 GB	5.58 GB	161.92 GB	307.08 GB	575.08 GB	3.58 TB	65.75
FED3R-FT $D = 10k$ Scaffold	77.20 MB	558.33 MB	635.53 MB	772.00 MB	5.58 GB	6.36 GB	190.66 GB	355.90 GB	667.30 GB	2.77 TB	67.99
FedNCM FedAvg	38.60 MB	48.98 MB	87.58 MB	386.00 MB	489.83 MB	875.83 MB	119.11 GB	423.03 GB	1.18 TB	-	53.72
FedNCM Scaffold	77.20 MB	87.58 MB	164.78 MB	772.00 MB	875.83 MB	1.65 GB	224.11 GB	576.74 GB	1.16 TB	-	56.60