Federated Least Squares: A Single Communication Round Algorithm for Federated Learning

Federico Borra, Emanuel Buttaci, Sara Montese

Polytechnic University of Turin
Turin, Italy

name.surname@studenti.polito.it

Abstract

Federated Learning (FL) is a distributed machine learning framework which jointly trains a global model via a large number of local devices with data privacy protections.
In this paper we take advantage of certain properties of convex problems and we apply them to the federated setting. In this paper, we propose a new approach for feature extraction, Rocket2D, based on the convolution of images with random kernels. We compare the results obtained with this method with a pre-trained model (VGG-19) used for feature extraction. In both cases we subsequently train a Ridge classifier, thus resulting in a convex problem. Then, we leverage the existence of a closed form solution to the L2 regularized Least Squares to solve the optimization problem in a single round of communication. Finally, we highlight some issues of the proposed approach and suggest some directions for future research.

1. Introduction

In this paper we are diving into the open problem of Federated Learning, an emerging framework designed to train a central machine learning model on multiple devices, without neglecting users' privacy and communication costs. Historically, Federated Learning was born to provide an efficient and decentralized way to train a server model on several users' devices with their local data. The importance of preserving the privacy of users is at the very core of Federated Learning, which avoids communication of private data to the server. Instead, the chosen approach trains a copy of the central model on users' devices and sends back to the server the trained weights of the model. The process is repeated for multiple rounds of the server, which chooses a subset of clients to be trained and then performs aggregation of their trained weights. As anticipated, there are several challenges associated with solving the distributed optimization problem posed above. These challenges set the federated setting apart from other conventional problems, such as traditional learning in data center settings.

Statistical Heterogeneity Due to diverse user behaviors, large heterogeneity may be present among different clients' local data. As a consequence, any particular user's local dataset will not be representative of the entire population distribution. This results in instability, slow convergence and a drop in the model performance.

Domain Shift Each client usually represents a different domain, which can vary in terms of data distribution, feature representations or other domain-specific characteristics. Thus, a model trained over several source domains may have poor generalization performance on unseen target domains.

Expensive Communication In the federated setting, communication methods must iteratively send small model updates as part of the training process. However, the total number of devices is typically massive, and this slows down communication due to limited computational resources. The aim is to reduce both the number of communication rounds and the size of the messages transmitted at each round.

Privacy Concerns Preserving privacy is the major concern. The learning procedure involves training models on data distributed across multiple devices, while ensuring the confidentiality and the protection of the user data. Nevertheless, finding a balance between privacy and model performance can be challenging, and improper implementation can lead to privacy breaches.

In this context, our goal in this work is to focus on these major challenges and to pursue improvements of the FL mechanism. Accordingly, our contributions are listed as follows:

- **Rocket2D**: a feature extractor based on the Rocket architecture for time series classification [5].
- **Federated Least Squares**: a federated algorithm that requires a single round of communication, and at the same time guarantees the same performance as the centralized version of least squares.

The first contribution helps in addressing the domain generalization problem, while the second tackles the communication and latency issues. Our code can be found at https://github.com/RicoBorra/FederatedLearningProject.

2. Related Work

The first relevant approach to Federated Learning is FedAvg [9], which simply aggregates trained parameters from clients by averaging them, weighted by their local datasets' sizes. Nowadays, FedAvg is often used as baseline for new FL algorithms. Originally, it proved its superiority by applying multiple local epochs of stochastic gradient descent on each client, rather than a single epoch, resulting in faster convergence and better test accuracy on the long run. Besides, by increasing local training epochs within each round of the server, communication costs are overall decreased. The objective is to minimize the global loss function $\mathcal{L}(\theta)$, where θ are the parameters of the central model:

$$\min_{\theta} \ \mathcal{L}(\theta) \triangleq \sum_{i=1}^{C} \frac{m_i}{m} \mathcal{L}_i(\theta)$$

We assume that data is distributed across C different clients, each one with m_i data samples available during the training, with m the total number of samples. During each server round t, each client applies E epochs of stochastic gradient descent to optimize its local loss $\mathcal{L}_i(\theta)$ over each client sample x_j, labeled as y_j:

$$\mathcal{L}_i(\theta) \triangleq \frac{1}{m_i} \sum_{j=1}^{m_i} l(x_j, y_j; \theta)$$

However, plain Federated Averaging could not properly face common issues of a realistic federated setting. First of all, data is not identically distributed among clients, a scenario commonly identified as *non-iid*. In this context, plain FedAvg with a higher number of local training epochs fails, since the central model deviates towards each client local minima, preventing the model from global convergence. In order to address this issue, known as client drift, FedProx [7] introduces a penalty term within the local loss $\mathcal{L}_i(\theta)$ of each client. This proximal term, weighted by μ, has the regularization effect of constraining the local weights θ not to deviate excessively from the central model's weights θ^t during local training.

$$\min_{\theta} \ \mathcal{L}_i(\theta) + \frac{\mu}{2} \|\theta - \theta^t\|^2$$

Concerning the heterogeneity among clients' data distributions, it is commonly hard to build a model capable of generalizing well over many different distributions. FedSR [10] directly faces this issue by learning a simple representation z of the client data x. Locally, the model is trained on the simple representation, derived from a probabilistic encoding. Two regularization terms are introduced in the local loss function, namely L2 penalty $\mathcal{L}_i^{\text{L2}}(\theta)$ and Kullback-Leibler divergence $\mathcal{L}_i^{\text{KL}}(\theta)$:

$$\mathcal{L}_i^{\text{L2}}(\theta) \triangleq \frac{1}{m_i} \sum_{j=1}^{m_i} \|z_j\|^2 \tag{1}$$

$$\mathcal{L}_i^{\text{KL}}(\theta) \triangleq \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbb{KL}(p(z|x_j) \,||\, r(z|y_j)) \tag{2}$$

With the help of these two terms, FedSR performs distribution alignment over the distributions $p(z|x)$ and $r(z|y)$, respectively modeled as multivariate Gaussians $\mathcal{N}_p(\mu(x), \text{diag}(\sigma(x)^2))$ and $\mathcal{N}_r(\mu(y), \text{diag}(\sigma(y)^2))$. The final local objective function of each client i is:

$$\min_{\theta} \ \widetilde{\mathcal{L}_i}(\theta) + \beta^{\text{L2}} \mathcal{L}_i^{\text{L2}}(\theta) + \beta^{\text{KL}} \mathcal{L}_i^{\text{KL}}(\theta)$$

where $\widetilde{\mathcal{L}_i}(\theta)$ is the objective function of the new representation z w.r.t. θ, and β^{L2} and β^{KL} are hyper-parameters. By learning a domain-invariant representation, it is possible to ignore domain-specific spurious correlations, leading to better generalization.

On the other hand, FedOpt [11] paper introduces different server optimization strategies to improve the global convergence of central model. Such technique imitates well-known optimization algorithms (i.e. Adam) to incorporate adaptivity in FL. In particular, the server updates its global model by applying a gradient-based server optimizer to establish a smarter aggregation of gradients received from C selected clients, at round t.

Moreover, client selection policy within each round also plays a substantial role on central model convergence. For instance, [2] approaches this issue by replacing uniform sampling of clients with a strategic approach referred to as *power-of-choice*. The idea is to choose, among a random subset of candidate clients, weighted by the amount of

data held, those clients revealing a higher local loss. This strategy has been proven to increase the rate of convergence compared to unbiased client selection.

3. Proposed Methods

The objective of our personal contribution is two-fold. Our first aim is to obtain a convex optimization problem. This ensures that, once reached a minimum, it is the only global minimum. Therefore, it hopefully generalizes better than the typical local minimum of a non-convex loss function. Secondly, we leverage the existence of a closed form solution to the L2 regularized Least Squares optimization problem, in order to drastically reduce latency.

3.1. Rocket2D

We propose a feature extraction technique based on the Rocket architecture used for time series classification [5]. The original architecture uses a high number of one-dimensional convolutional kernels (10,000 by default) with random length, weights, bias, dilation and padding. The authors introduce a new pooling mechanism called **PPV**: *percentage of positive values*. A conventional MaxPooling is also used. Therefore, two features are extracted for each kernel: one obtained through PPV and one obtained through MaxPooling. The extracted features are then used to train a Ridge classifier, thus resulting in a convex optimization problem.

The PPV pooling mechanism is of paramount importance in the high accuracy obtained by Rocket. Since the authors show that using only PPV does not cause a meaningful drop in accuracy, we keep it as our sole pooling mechanism in our bi-dimensional version of Rocket.

Considering that the original architecture uses one-dimensional kernels, we redesign it in order to make it compatible with bi-dimensional inputs. We use the same distributions of the paper both for row and column dilation and we assume statistical independence of the two. We proceed in an analogous manner for the weight distribution and also for the row length and columns length. These last parameters are chosen with uniform probability from three values which are approximately equal to the square root of the (single) length that was present in the Rocket paper: this is done to keep the kernels as small as possible since the authors use kernels lengths that are considerably smaller than the time series to be classified. However, with lower lengths comes an increased chance of the kernels being highly correlated with each other. This choice of lengths would therefore warrant further experimentation and testing, especially taking into account the considerations we will make in Subsubsection 4.6.2.

3.2. Federated Least Squares

We propose a new algorithm called Federated Least Squares, or **FedLSQ** for short. This algorithm allows us to optimize an L2 regularized Least Squares problem with a closed form, using a single round of communication from the clients to the server. This new approach also guarantees a solution that is the same as the one that would have been obtained in a centralized scenario (save for numerical stability considerations).
The crucial point we want to highlight is that solving a Least Squares problem with L2 penalty

$$\min_B ||Y - XB||^2 + \lambda||B||^2$$

is equivalent to solving the following linear system with respect to B:

$$(X^TX + \lambda I)B = X^TY \tag{3}$$

where X is the design matrix, B is the vector of coefficients of the linear model and Y is the label vector. This result can be proven with basic matrix differentiation techniques, leveraging the strict convexity and smoothness of the problem, which implies that the only critical point of the cost function (if such a point exists) is the global minimum.
Both matrix X^TX and vector X^TY are comprised of elements which are a sum of products. These products' factors come from the same row and possibly different columns of either X or Y. The index of these sums is in all cases relative to the rows of X and Y. A more detailed explanation can be found in the Appendix. Therefore, by the associativity and commutativity properties of the sum we can write

$$X^TX = \sum_{i=1}^{C} X_i^TX_i$$

and

$$X^TY = \sum_{i=1}^{C} X_i^TY_i$$

where X_i and Y_i are, respectively, the design matrix owned by client i and its label vector, with C being the total number of clients. By aggregating these partial sums that are sent from the clients, the server can therefore obtain the same solution that would have been found if the dataset was in the server in its entirety.
A possible drawback of this approach is that if we know the number of rows of X we can recover $\tilde{X}$ by applying SVD to X^TX, where $X = A\tilde{X}$ and A is an isometry. Depending on how computationally difficult it is to find the correct isometry A, it could be feasible to recover the design matrix X_i from the local updates $X_i^TX_i$. Other de-anonymization techniques could be introduced by combining $X_i^TX_i$ with

$X_i^T Y_i$. Further research on this topic would be needed before adoption of this algorithm in settings where sensitive data is handled. In particular, we believe the differential privacy framework to be promising in this case.

However, in our case the design matrices X_i are feature maps of the original datasets present in the clients. Moreover, the server does not know the number of samples stored in each client and would therefore need to use a brute-force approach to the first de-anonymization technique described above. Thus, we consider ourselves satisfied with the current privacy protection for the time being.

Lastly, we highlight the on-line learning capabilities of this method: a naive but effective approach would be to store the $X^T X$ matrix as an internal state before computing the optimal solution. When other local updates arrive, we sum them to the state matrix. Then, we only need to re-compute the solution to the linear system described in Eq. 3 to obtain an optimal solution. This solution takes into account the new data, while at the same time performing as well as it would have done if the new samples were in the original training set all along [13].

4. Experimental Results

4.1. Datasets and Experimental Setting

Datasets In this work, the experiments are performed on two datasets. EMNIST (Extended MNIST) [4] is a handwritten digits dataset with 814,255 images (80% for training, 20% for testing) from 62 classes. The data points consist of 28x28 gray-scale images representing digits from 0 to 9, 26 lowercase and 26 uppercase characters of the English alphabet. To start with, the model is trained on this dataset in a centralized way. The obtained results represent an upper bound for the performance of any model later trained in the federated setting. The second dataset used is FEMNIST (Federated Extended MNIST) [1], a benchmark dataset for FL built by partitioning the data in EMNIST based on the writer of the symbol. The dataset contains 817,851 images (80% for training, 20% for testing) from 62 classes, and it is spread across 3,597 clients, as shown in Figure 1.

Both a uniform and a heterogeneous version of the federated dataset are supported. In the uniform distribution (*i.i.d.*), each client has access to a subset of the images drawn randomly from the whole dataset. Since such distribution does not consider the real-world challenge of statistical heterogeneity, we only use it as a reference for the model's performance. On the other hand, the heterogeneous federated version of FEMNIST assumes that the data distribution varies between users. In this scenario, referred to as *non-i.i.d.*, each client is a different writer, so the local datasets differ on both classes and handwriting. To test the generalization capabilities of the model, we set up several test clients that contain images never seen at training time, for both the *i.i.d.* and *non-i.i.d.* settings.

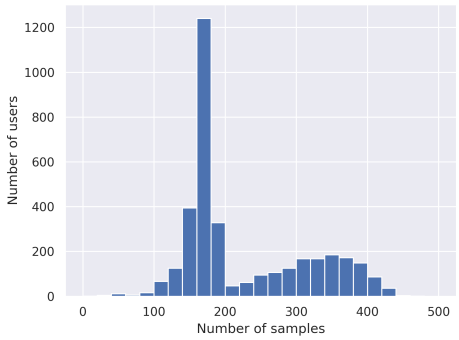

Figure 1. Samples distribution across users in FEMNIST

Preprocessing In order to enhance the computational efficiency on a large dataset like FEMNIST, we adopted our own pipeline for loading and dealing with data within each experiment. Particularly, the FEMNIST dataset is now generated from a pipeline of scripts which outputs the training and testing files, both in *i.i.d.* and *non-i.i.d.* cases, as *parquet* files. This major change gives the possibility of loading the entire data in RAM in less than five seconds, compared to around ten minutes required with the *json* version of the dataset. In addition, we reduce the occupied space of the original *json* dataset on disk from roughly 4 GB to less than 500 MB. Hopefully, our pipeline will be adopted for further experimentation on the FEMNIST dataset in the near future.

Models The network architecture used in the experiments is showed in Table 1. The network is a CNN with two 5x5 convolutional layers, with 32 and 64 output channels respectively. Each convolutional layer is followed by both a 2x2 max pooling layer and a ReLU activation layer. Finally, we add two fully connected layers, the first one mapping the input to 2048 output features, and the second one mapping the features to the number of classes.

Validation metrics Given $K = 62$ classes, we adopt two validation metrics to assess the performance of our model, in such a way to capture the sensibility to the unbalancing of the classes.

$$\textbf{UA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N}}$$

$$\textbf{WA} = \frac{1}{K} \sum_{k=1}^{K} \frac{\mathrm{TP}_k + \mathrm{TN}_k}{\mathrm{N}_k}$$

Specifically, **UA** is the overall accuracy over all N samples, whilst **WA** is the average of the single class accuracies for

Table 1. CNN model architecture

Layer	Output Shape	Activation	Hyper-parameters
Input	(28,28,1)		
Conv2d	(28,28,32)		kernel_size = 5 stride = 1 padding = 'same'
MaxPool2d	(14,14,32)	ReLU	pool_size = 2 stride = 2
Conv2d	(14,14,64)		kernel_size = 5 stride = 1 padding = 'same'
MaxPool2d	(7,7,64)	ReLU	pool_size = 2 stride = 2
Flatten	3136		
Linear	2048	ReLU	
Linear	62		

each class k.

Table 2. FEMNIST dataset partition

Group	Dataset %
Training	64 %
Validation	16 %
Testing	20 %

Finally, our clients are divided as depicted in Table 2. We choose to hold out a validation set of disjoint clients from the training set in order to have a more realistic understanding of the model generalization performance.

4.2. Federated Scenario Baseline

To define an upper bound for the experiments in the federated setting, we firstly train the model (1) on EMNIST. The network is trained for 10 epochs, with stochastic gradient descent, using a learning rate of 0.005 decaying with a factor of 0.5 every epoch, L2 penalty of 10^{-5}, momentum of 0.9 and batch size of 256. The obtained test accuracy of 86.74 ± 0.01 % is used as baseline for the centralized setting.

To define a baseline for the federated scenario, we train and validate the model in Table 1 using different combinations of hyper-parameters. Given E as the number of local epochs, C as the number of clients in each epoch, Tables 3 and 4 summarize the weighted and the overall accuracies obtained in the various configurations. The network is validated using a learning rate of 0.05 decaying of a factor of 0.75 every 50 epochs, L2 penalty of 10^{-5}, momentum of 0.9 and batch size of 64.

Tables 3 and 4 show the impact of the different number of epochs and clients. Particularly, increasing the number of

Table 3. FedAvg baseline on IID FEMNIST

E	C	WA	UA
1	5	67.00 ± 0.72	82.05 ± 0.75
	10	67.62 ± 1.28	82.54 ± 2.09
	20	68.14 ± 0.39	82.88 ± 1.17
5	5	74.18 ± 1.41	85.91 ± 1.30
	10	**75.21 ± 1.44**	**86.23 ± 1.60**
	20	75.69 ± 0.40	86.61 ± 0.66
10	5	74.07 ± 1.61	85.85 ± 1.03
	10	75.29 ± 1.31	86.14 ± 1.15
	20	75.72 ± 0.62	86.68 ± 0.54

Table 4. FedAvg baseline on NIID FEMNIST

E	C	WA	UA
1	5	68.03 ± 0.11	82.19 ± 0.52
	10	68.44 ± 1.45	81.80 ± 1.92
	20	69.03 ± 0.30	82.41 ± 0.80
5	5	75.77 ± 0.60	85.59 ± 0.37
	10	**76.25 ± 2.28**	**85.22 ± 1.30**
	20	76.73 ± 0.74	86.10 ± 0.59
10	5	76.05 ± 0.45	85.59 ± 0.38
	10	76.56 ± 1.90	85.43 ± 1.39
	20	76.87 ± 0.54	86.21 ± 0.40

local epochs has a higher positive impact on the overall accuracy with respect to the choice of the number of clients involved in the training round. Indeed, by using a single local epoch we get lower results, as expected from FedSGD. However, the choice of E must be reasonable in order to avoid the performance degradation mainly attributed to the divergence of the local losses. This is caused by the fact that each client approaches its local loss minimum, which does not necessarily corresponds to the global minimum. Taking into account the trade-off between accuracy and computational effort, the chosen configuration of hyper-parameters involves running 5 local epochs of training over 10 clients chosen at each round.

4.3. Smart Client Selection

Choosing clients in a strategic way can result in better performance in terms of convergence speed and accuracy. Among the possible choices regarding client selection criteria, we employ a random hybrid approach and the power-of-choice method described in [2]. The hybrid approach partitions the training clients in two subsets, different in sizes. Each partition is characterized by a certain probability for each client of being selected during the training rounds. The number of local epochs E is fixed to 5 for these experiments.

Table 5 shows the comparison in term of accuracy obtained using different client selection criteria. It is evident from

Figure 2 the higher rate of convergence proved by power-of-choice criterion against uniform client selection.

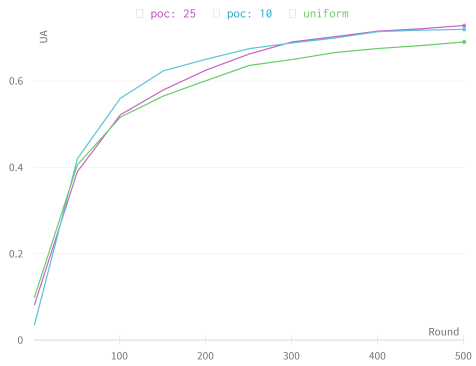

Figure 2. Validation accuracy with strategic clients selection

Table 5. Different selection criteria

IID	C	Selection	WA	UA
Yes	5	uniform	74.18	85.91
		poc: 10	74.77	86.09
		poc: 25	**73.95**	**86.42**
No	5	uniform	75.77	85.59
		poc: 10	75.46	85.48
		poc: 25	**75.06**	**86.00**

As evident from the same table, the choice of the number of candidates on which the loss is evaluated has an impact on the overall score. Thus, it becomes easier with a larger candidates set to find those clients having a worse performance in terms of local loss, even though this comes at a heavier computational cost.

4.4. Rotated Domain Generalization

As introduced in Paragraph 1, the domain shift represents one of the main challenges in FL. In order to make the model generalize well on unseen target domains, in the experiments we follow the practice in domain generalization literature to adopt the leave-one-domain-out strategy, i.e., training on K-1 source domains and testing on the left-out unseen target domain. Based on this practice, we build a new version of the dataset to introduce new domains by means of data augmentation. The new dataset Rotated-FEMNIST is generated by rotating images counterclockwise with an angle of 0°, 15°, 30°, 45°, 60° and 75° to form six different domains, respectively $\mathcal{M}_0$, $\mathcal{M}_{15}$, $\mathcal{M}_{30}$, $\mathcal{M}_{45}$, $\mathcal{M}_{60}$ and $\mathcal{M}_{75}$. We run the first experiment using an *Overall* strategy without the leave-one-out approach for validation. Then, we evaluate each single rotated domain in a leave-one-out fashion across six runs. For these experiments we compare FedAvg and FedSR behaviours, fixing local epochs E to 5 and chosen clients C to 10. Specifically, FedSR has been tested using dimensionality equal to 128 for the simple representation z and regularization coefficients $\beta^{\text{L2}} = 0.1$ and $\beta^{\text{KL}} = 0.1$. Moreover, FedSR has been implemented based on a probabilistic encoding architecture in order to make use of the regularization term $\mathcal{L}^{\text{KL}}(\theta)$ defined in Eq. 2. As shown in Table 11, the chosen configuration of hyper-parameters for FedSR could not prove better generalization scores when compared with FedAvg.

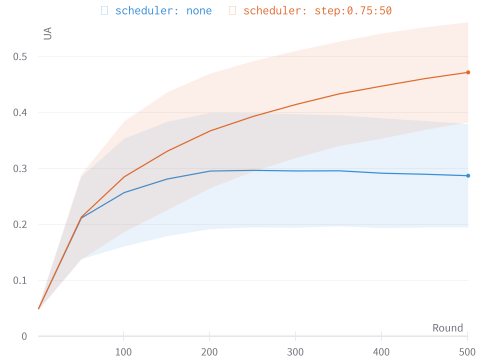

Figure 3. FedSR sensitivity to learning rate scheduling

Although, in Figure 3 it is evident how FedSR is susceptible to the presence of the learning rate scheduler. From our experimentation, FedAvg does not suffer from this behaviour, while FedSR manifests a dramatic drop in validation accuracy, roughly 20% on average, when the scheduler is not employed during local optimization. Again, as scheduling strategy a decay of a factor of 0.75 every 50 central rounds has been used.
Therefore, we believe that a proper choice of hyper-parameters regarding FedSR could provide promising results when compared to plain FedAvg. As mentioned before, the choice of the learning rate scheduling plays a decisive role in the overall result.

4.5. Optimization Strategies

FedYogi FedYogi is one of the algorithm proposed in paper [11] in order to improve server-side optimization with respect to original FedAvg. Differently from other algorithms, FedYogi introduces many parameters to be tuned in order to achieve comparable performance. Among these, we keep fixed the number of local epochs to 5 and the amount of chosen clients to 10, but also $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\tau = 0.0001$. Also, for the sake of simplicity, we set the learning rate of clients to 0.05 with decay as in other experiments. Note that only the *non-i.i.d.* configuration has been used to assess the generalization capability of the algorithm.

Table 6. Comparison of FedYogi and FedAvg

E	C	Algorithm	WA	UA
		FedAvg	76.25	85.22
5	10	FedYogi: $\eta = 0.001$	75.42	84.73
		FedYogi: $\eta = 0.01$	71.77	83.67

From Table 6 it is evident how FedYogi performance is affected by the order of magnitude of the server learning rate η. Specifically, according to the original paper, the server learning rate should be inversely related to the learning rate of the clients.

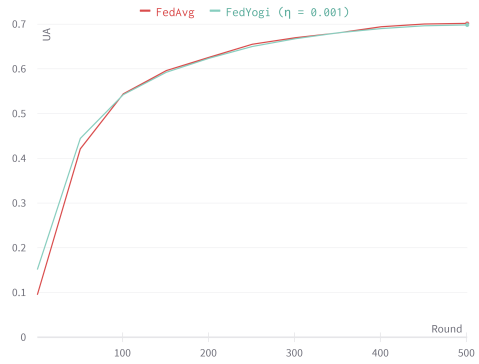

Figure 4. Validation accuracy of FedYogi and FedAvg

Figure 4 shows the validation overall accuracy of both FedAvg and FedYogi with $\eta = 0.001$. We deduce that particular configurations of hyper-parameters for FedYogi could slightly increase the initial speed of convergence when compared with plain FedAvg.

FedProx Another competitive algorithm in the context of clients' domain shift is FedProx [7]. It has been tested against FedAvg while keeping the same experimental setting depicted before. Furthermore, we assess the performance of the dynamic version of FedProx, namely having a variable parameter μ. In our version, we increase μ of a factor of 1.5 every 100 server rounds.

From the validation curves in Figure 5 we can observe that a dynamic scheduling of μ brings benefits to the performance on unseen clients (validation mode). Thus, we strongly believe that a proper choice of such scheduling function of μ could eventually carry improvements on the long run. In particular, an idea would be to use a scheduling function that increases μ on loss plateau, decreases μ on loss worsening and stays stable when the loss is improving.

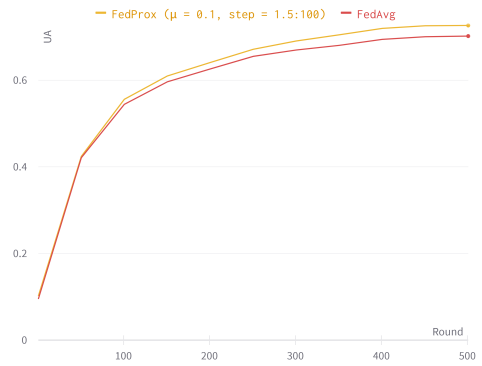

Figure 5. Validation accuracy of FedProx and FedAvg

4.6. Proposed solutions

4.6.1 Preliminary Considerations

The results in this section are obtained using only 10% of the original dataset, due to the lack of computational resources as well as time available for the runs. In order to get comparable results, we re-train on this reduced dataset the CNN based model using FedAvg with the best hyper-parameters found in Section 4.2, namely 10 local epochs and 20 clients selected per round.
We compare the results obtained with the CNN with the ones coming from Rocket2D and from a pre-trained VGG [12] used as feature extractors. The feature map is fed to by a Ridge classifier, in both cases trained with FedLSQ. Note that the convergence of a smooth strongly convex optimization problem, such as Ridge, to its single global minimum has already been proven even in federated settings (on the condition of using a decaying learning rate) [8]. Therefore, we only focus on FedLSQ for the Ridge optimization. For the Ridge classifier the regularization hyper-parameter λ is fixed to 1.

4.6.2 Condition Number and PCA

Table 7. Comparison of condition numbers

Feature Extractor	Type	X^TX	$X^TX + I$
Rocket2D	IID	∞	$9.0 \cdot 10^{11}$
	NIID	∞	$8.6 \cdot 10^{11}$
Rocket2D + PCA	IID	583.6	583.5
	NIID	1068.1	1067.7
VGG	IID	$2.8 \cdot 10^{21}$	$1.9 \cdot 10^{9}$
	NIID	$4.4 \cdot 10^{19}$	$5.1 \cdot 10^{8}$
VGG + PCA	IID	790.1	789.9
	NIID	963.2	963.0

The condition number of the matrices X^TX obtained after

using the feature extractors is particularly high (as shown in Table 7), and in some cases it overflows (indicating a non-invertible matrix). This is a thorny problem since the condition number of matrix A measures how accurate the solution to the linear system $AX = B$ is. We hypothesize that this problem is rendering some results unusable (in particular the ones using Rocket2D as a feature extractor).
One possible solution to this problem is to make use of Tikhonov regularization [3], which is particularly handy since in our case we already need to perform such an operation for Ridge regression (i.e. adding λI to $X^T X$). If the previous proposal is not sufficient we could resort to PCA. It is known that a high condition number is a symptom of multicollinearity among the columns of a matrix. Since PCA yields the projection of the feature vectors on a set of orthogonal components, applying this technique minimizes the multicollinearity of the data.
In an infinite precision environment we would expect the composition of PCA (which is a linear transformation) and the linear trasformation operated by Ridge to give an upper bound on the minimum of the objective function given by Ridge alone. Our experiments show that this is not the case when using Rocket2D as feature extractor, while being true when using VGG. This corroborates our hypothesis. There is an established algorithm that can be used to perform a federated version of PCA [6], but in this work we simply assume that the server has access to a subset of the data. This enables the server to perform PCA on the stored subset of the dataset. This way the server can communicate the projection matrix on the first n components together with the centralized model. We choose a number of components which is 90% of the features.

4.6.3 Results

The combination of the proposed methods, **Rocket2D** and **FedLSQ**, allows us to achieve three important results, the first two are summarized in Table 8.

- We are able to get an accuracy on new samples from known clients (**Known UA**) very similar to the one obtained with the CNN trained with FedAvg.
- We perform better on domain generalization: we obtain a higher accuracy on samples from unseen clients (**Unseen UA**) compared to the FedAvg trained CNN.
- We achieve a 500x reduction in latency time while having a comparable total bandwidth consumption in the case of VGG. A more detailed analysis is to be found in the Appendix.

We highlight that further reduction in the total bandwidth consumed could be made possible by either reducing the number of clients participating or by keeping a lower number of principal components. This last option would yield very good results especially considering that the total amount of data exchanged is $O(Cf^2)$ where f is the number of features we keep and C is the number of clients. In Table 8 is also worth mentioning the comparison between Rocket2D followed by PCA with VGG-19 pre-trained on ImageNet, both used as feature extractors. We can see that they achieve very similar performances, thus confirming the value of Rocket2D as a feature extractor, especially considering that it does not need pre-training.
The usage of known clients (testing set) versus unseen clients (validation set) to measure the domain generalization capability of the model was chosen for two reasons. Firstly, the natural domain shift occurring among the various clients is more representative of a real case scenario. Secondly, the shortage of time at our disposal and the lack of access to computational resources made very difficult the repetition of the procedure followed in Subsection 4.4. The split follows the methodology described at the end of Subsection 4.1

Table 8. Comparison of different architectures on FEMNIST

IID	Algorithm	Model	Unseen UA	Known UA
Yes	FedAvg	CNN	74.16	82.21
	FedLSQ	Rocket2D + PCA	81.25	80.51
	FedLSQ	VGG	82.22	81.94
No	FedAvg	CNN	73.91	83.37
	FedLSQ	Rocket + PCA	81.27	80.84
	FedLSQ	VGG	81.52	82.52

5. Discussion

In this paper we prove the viability of using a model based on feature extraction followed by a Ridge classifier in a Federated Learning setting.
We introduce a new feature extractor, Rocket2D, which performs very similarly to a pre-trained model like VGG, while not needing a pre-training phase.
We also show that it generalizes better than its CNN counterpart on unseen domains.
Finally, by using our novel optimization algorithm FedLSQ, we manage to train the model optimally (i.e. as would have happened in a centralized scenario) at a fraction of the latency.

References

[1] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings, 2018. 4

[2] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, 2020. 2, 5

[3] Delin Chu, Lijing Lin, Roger C. E. Tan, and Yimin Wei. Condition numbers and perturbation analysis for the tikhonov regularization of discrete ill-posed problems. *Numerical Linear Algebra with Applications*, 18(1):87–103, Feb. 2010. 8

[4] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters, 2017. 4

[5] Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, jul 2020. 2, 3

[6] Andreas Grammenos, Rodrigo Mendoza-Smith, Jon Crowcroft, and Cecilia Mascolo. Federated principal component analysis, 2020. 8

[7] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020. 2, 7

[8] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data, 2020. 7

[9] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023. 2

[10] A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. FedSR: A simple and effective domain generalization method for federated learning, 2022. 2

[11] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2020. 2, 6

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 7

[13] Ruohan Wang, Marco Ciccone, Giulia Luise, Andrew Yapp, Massimiliano Pontil, and Carlo Ciliberto. Schedule-robust online continual learning, 2022. 4

Appendix

Federated Least Squares

Let us expand upon the claim that the elements from the matrix X^TX can be reconstructed from the sum of the elements of the matrices $X_i^TX_i$ of each client i (and similarly for vector X^TY). Note that here we are considering a Ridge regression on a single target variable. This is not reductive since in a one-vs-rest ensemble model the following reasoning would apply to all the single class Ridges and therefore to the model as a whole.

Let us define x_{ij} as the element at row i and column j of the matrix X. Let us suppose that matrix X has n rows and f columns. Moreover, X is divided into C sub-matrices X_i of n_i rows and f columns stacked on top of each other. We suppose that client i owns the corresponding matrix X_i.

The situation is such that

$$X^T = \left[X_1^T | X_2^T | \dots | X_C^T\right]$$

and of course

$$\sum_{i=1}^{C} n_i = n$$

Now, let us define G to be the gram matrix X^TX. Then element

$$g_{ab} = \sum_{i=1}^{n} x_{ia}x_{ib}$$

By associativity of the sum it is clear that

$$g_{ab} = \sum_{i=1}^{n_1} x_{ia}x_{ib} + \sum_{i=n1+1}^{n1+n2} x_{ia}x_{ib} + \dots$$

where every term is comprised of a sum of products of elements all belonging to a given submatrix (i.e. the first term belongs to submatrix X_1 and so on). This applies to all elements of the gram matrix X^TX as well as to all elements of the vector X^TY.

If various clients possess the matrices X_i then by communicating the local updates $X_i^TX_i$ the central server just need to compute their sum in order to obtain the complete matrix X^TX. Since the addition is commutative, it does not matter in which order we perform these operations, we will always reach the same end result.

Experimental results

In Table 9, we compare the performances obtained by the power-of-choice client selection approach, by varying the number of the candidate clients and the number of truly trained clients. We notice that having a larger pool of candidates to select from typically benefits the model performance. The results obtained by applying the hybrid client selection criterion are reported in Table 10.

In Table 11 we compare the validation performance of FedAvg and FedSR on all rotated domains. As it can be seen, accuracy degrades as rotation angle increases. In the case of *Overall* scenario no specific rotated domain is used for validation, instead evaluation is performed with both original and transformed (by rotation) clients.

Regarding FedProx, in Table 12 we present results using both a static and dynamic hyper-parameter μ.

Communication efficiency analysis

We derive some conclusions With regard to the communication costs involved in our experiments on federated algorithms. Specifically, in Table 13 we compare the CNN model (used in combination with FedAvg) to our Ridge (trained on transformed FEMNIST with FedLSQ).

Table 9. FedAvg with power-of-choice client selection

IID	Candidates	C	WA	UA
Yes	10	2	73.44	85.77
	10	5	74.77	86.09
	10	8	75.06	85.51
	25	2	73.57	86.02
	25	5	73.95	86.42
	25	8	74.31	85.67
No	10	2	72.33	85.01
	10	5	75.46	85.48
	10	8	75.96	84.96
	25	2	72.73	84.78
	25	5	75.06	86.00
	25	8	75.94	86.30

Table 10. FedAvg with hybrid client selection

IID	Fraction	Probability	WA	UA
Yes	0.0001	0.3	75.01	86.50
	0.5	0.1	74.53	85.98
No	0.0001	0.3	76.27	86.19
	0.5	0.1	74.85	84.96

Table 11. FedAvg performance on RotatedFEMNIST

Domain	Centralized	FedAvg	FedSR
Overall	88.76	66.61	60.68
$\mathcal{M}_0$	72.04	74.28	52.49
$\mathcal{M}_{15}$	70.64	69.02	52.47
$\mathcal{M}_{30}$	70.78	63.52	50.93
$\mathcal{M}_{45}$	72.78	62.96	52.37
$\mathcal{M}_{60}$	70.38	61.95	44.76
$\mathcal{M}_{75}$	59.30	42.66	29.96

Table 12. Comparison between FedProx and FedAvg

E	C	Algorithm	WA	UA
5	10	FedAvg	76.25	85.22
		FedProx: $\mu = 0.01$	75.73	84.79
		FedProx: $\mu = 0.1$	75.28	84.53
		FedProx: $\mu = 0.1$ (dynamic)	75.37	84.68

Table 13. Comparison of communication costs

Algorithm	Model	Features	Space [MB]	Bandwidth [GB]
FedAvg	CNN	6603710	26.4	264.1
FedLSQ	VGG	17031168	68.1	233.0
FedLSQ	Rocket2D	200000000	800.0	2736.0

We choose these metrics which are considered more impactful on a communication basis. Regarding FedAvg

$$\text{Space}_{\text{FedAvg}} \triangleq \text{sizeof(float32)} \cdot \text{F}_{\text{CNN}}$$
$$\text{Bandwidth}_{\text{FedAvg}} \triangleq 2 \cdot \text{R} \cdot \text{C} \cdot \text{Space}_{\text{FedAvg}}$$

where $R = 500$ is the number of total rounds, $C = 20$ is the number of selected clients and F is the number of trainable parameters. On the other hand, computation for FedLSQ is independent from the number of rounds, thus

$$\text{Space}_{\text{FedLSQ}} \triangleq \text{sizeof(float32)} \cdot (\text{F}^2_{\text{Ridge}} + \text{F}_{\text{Ridge}} \cdot K)$$
$$\text{Bandwidth}_{\text{FedLSQ}} \triangleq \text{C} \cdot \text{Space}_{\text{FedLSQ}}$$

In this case F_{Ridge} is 4096 and 10000, respectively for VGG and Rocket2D and $C = 3600$ is the number of total clients involved in the single communication round expected by FedLSQ. $K = 62$ denotes the number of classes.