mathematics

MDPI

Article

A Heterogeneous Federated Transfer Learning Approach with Extreme Aggregation and Speed

Tarek Berghout [1], Toufik Bentrcia [1], Mohamed Amine Ferrag [2,*] and Mohamed Benbouzid [3,4]

[1] Laboratory of Automation and Manufacturing Engineering, University of Batna 2, Batna 05000, Algeria
[2] Department of Computer Science, University of Guelma, Guelma 24000, Algeria
[3] Institut de Recherche Dupuy de Lôme (UMR CNRS 6027), University of Brest, 29238 Brest, France
[4] Logistics Engineering College, Shanghai Maritime University, Shanghai 201306, China
* Correspondence: ferrag.mohamedamine@univ-guelma.dz

Abstract: Federated learning (FL) is a data-privacy-preserving, decentralized process that allows local edge devices of smart infrastructures to train a collaborative model independently while keeping data localized. FL algorithms, encompassing a well-structured average of the training parameters (e.g., the weights and biases resulting from training-based stochastic gradient descent variants), are subject to many challenges, namely expensive communication, systems heterogeneity, statistical heterogeneity, and privacy concerns. In this context, our paper targets the four aforementioned challenges while focusing on reducing communication and computational costs by involving recursive least squares (RLS) training rules. Accordingly, to the best of our knowledge, this is the first time that the RLS algorithm is modified to completely accommodate non-independent and identically distributed data (non-IID) for federated transfer learning (FTL). Furthermore, this paper also introduces a newly generated dataset capable of emulating such real conditions and of making data investigation available on ordinary commercial computers with quad-core microprocessors and less need for higher computing hardware. Applications of FTL-RLS on the generated data under different levels of complexity closely related to different levels of cardinality lead to a variety of conclusions supporting its performance for future uses.

Keywords: federated learning; federated transfer learning; heterogeneous systems; non identical independent data; recursive least squares

MSC: 93E24

Citation: Berghout, T.; Bentrcia, T.; Ferrag, M.A.; Benbouzid, M. A Heterogeneous Federated Transfer Learning Approach with Extreme Aggregation and Speed. *Mathematics* **2022**, *10*, 3528. https://doi.org/10.3390/math10193528

Academic Editor: Antanas Cenys

Received: 4 September 2022
Accepted: 24 September 2022
Published: 28 September 2022

1. Introduction

Federated learning (FL) is the process of aggregating a set of machine learning (ML) models related to a specific set of remote sensors or isolated data centers while keeping data as private as possible [1]. FL faces many challenges related to expensive communications, systems heterogeneity, statistical heterogeneity, and privacy concerns, respectively. Dealing with such a situation requires a fundamental departure from traditional centralized/decentralized ML toward distributed optimization and privacy preservation [2]. When considering data partitioning as the main criterion of FL models classification, FL can then be classified into three main categories, specifically, horizontal, vertical, and federated transfer learning (FTL) [3]. As a result, the evolution of FL procedures while aggregating local models mainly focuses on these challenges. This gives rise to many types of algorithms derived from variants of stochastic gradient descent such as federated averaging (FedAvg) and federated stochastic variance reduced gradient (FSVRG) [4]. These models were also designed to solve model synchronization problems when transferring local models for global updates.

In this context, many algorithms have been published recently. For example in the context of targeting expensive communication problems, the authors in [5] succeeded

in developing a deep learning (DL) communication compression scheme for large-scale data applicable to decentralized training without data sharing. Their study shows that the proposed method applies to both IID and non-IID data. They developed a consensus algorithm to adapt the training of the global model to a heavy communication load and to avoid communication-blocking problems caused by some algorithms such as FedAvg or FSVRG. For evaluation purposes, the MNIST [6] handwritten digital image dataset was adopted in this case where the algorithm shows superior performance. In [7], the authors managed to overcome FL communication and privacy issues resulting from limited network bandwidth and advanced privacy attacks by developing an effective privacy preservation and communication scheme. Their main contributions to expansive communication problems have been the introduction of bi-directional compression, where computationless compression operators are used to quantify gradients on both global and local model sides. Two real-world datasets, namely the MNIST [6] and a dataset created from the comprehensive work of Shakespeare [8], are involved in assessing the accuracy of the learning process. To overcome the higher level of communication resource consumption caused by FL decentralized learning paradigms as well as privacy issues, a new compression algorithm based on sparse coding for efficient communication, with additive holomorphic encryption with a differential privacy to prevent data leakage, is proposed in [9]. The compression philosophy has led to both effective communication and the preservation of privacy. Experimental evaluation on the MNIST [6] dataset has demonstrated that the proposed algorithm outperforms basic FL approaches including algorithms such as FedAvg and Paillier-encryption-based privacy-preserving deep learning (PPDL) on many criteria while exhibiting good convergence features.

Under circumstances of system heterogeneity, sets of successful recent FL works are chosen as examples with the aim of circumventing its problems. In [10], an asynchronous FL scheme has been constructed to solve the problems of local learning models in terms of device heterogeneity and unstable communication. Thus, an iterative selection algorithm is adopted to efficiently manage the learning tasks by taking into account the non-synchronization of the delivered messages. Many experiments on IID and non-IID distributions obtained from several datasets, namely the MNIST dataset [6], fashion-MNIST dataset [11] and EMNIST dataset [12], have revealed the effectiveness of the proposed scheme. In [13], the authors investigated the use of hierarchical FL in heterogeneous systems by introducing an optimized solution for user assignment and resource allocation. In their work, they paid more attention to variants based on gradient descent while taking into account the imbalanced data between different users. Their training methodology is tested on two real-world datasets, namely the Heartbeat dataset [14] and the Seizure dataset [15], and shows that it outperforms existing FL methods. In [16], a semi-supervised FL framework was proposed for heterogeneous transfer learning to take advantage of unlabeled non-overlapping samples and to help reduce overfitting due to insufficient overlapping training samples. Through extensive experiments based on the IMDB [17], Handwritten [18] and ALLAML [19] datasets, the designed scheme showcased significant advantages over state-of-the-art approaches.

Concerning statistical heterogeneity, recent FL works are focusing on learning global models under non-IID. This is can be justified by the non-identically used devices for different tasks, e.g., mobile phone users around the globe use different languages making, for instance, text/speech recognition a difficult task. In [20], a FL learning approach that deals with this issue is introduced. Similarity between clients in the context of retrieved data distribution is used to model their relationships. This methodology, however, attempts to determine this relationship without accessing the data, which violate privacy concerns in FL. In this context, a similarity-based algorithm (FedSim) was introduced to decompose model aggregation into local and global steps. Within this framework, clients with similar gradients are considered for the aggregation process, while later they can be globally aggregated to provide better coverage jointly with the reduction of gradient variance. Using real world datasets including MNIST [6], Fed-MEx [21], and Fed-Goodreads [22], and compared

to well-known FL algorithms, such as FedAvg and FedProx, FedSim emphasizes the significance of these contributions. In [23], the authors proposed training in heterogeneous model aggregation (MHAT) to be able to solve local model problems containing various network architectures. Accordingly, model aggregation was achieved by exploiting knowledge distillation (KD) techniques for information extraction and update while training an auxiliary model. Various experiments on the MNIST dataset [1] have confirmed the effectiveness of the proposed scheme. In [24], the authors proposed a mobile application startup prediction model based on FL under heterogeneous network integration, which solves the cold-start problem of new users or new applications while ensuring user privacy. Experiments were done on a LiveLab dataset while their application showed promising performance.

Regarding privacy concerns, an FL approach is proposed in [25] to solve the cold-start problem of elements in recommender systems. The work contemplates proposing a trust-based mechanism for potential recommenders. Extensive reinforcement learning planning was used to select the best candidates. Experiments on MovieLens [26] and Epinions [27] datasets have highlighted the trainability of the model both in terms of accuracy and computation time. In [28], a privacy-preserving FL and non-interactive gradient descent (VANE) were proposed, while the aggregation process can be performed without disclosing private information or any interaction between clients. Performance evaluation results on multiple real-world datasets demonstrated that the performances of VANE is at a higher level of speed: approximately 103 times faster than existing schemes. The authors in [29] introduced a privacy-preserving FL with DL processes based on the trusted execution environment (TEE). Using datasets such as the MNIST dataset [6], model performance evaluation results highlighted that the schema is practical and ensures training integrity. For recapitulation, Table 1 is introduced to summarize the above-cited works in a clear and effective way by presenting the used datasets besides the main contributions to solve each of the four FL challenges.

Results of the literature review provided in Table 1 lead to important conclusions:

1. In terms of FL challenges and more specifically "expansive commutation" related to the number of transmitted messages and the size of the transmitted information of the training models, these models follow different lossy compression schemes to reduce the size of the model architecture in terms of the number of bits. However, these models, being DL networks, have a serious disadvantage of being large models with multiple parameters;
2. As these models are trained with gradient descent algorithms which are iterative algorithms and are subject to a higher level of computational costs during training, the training process will be very expansive and time consuming when it acts to rebuild the feature map, and will also take time when it comes to aggregation;
3. In terms of "system heterogeneity", another disadvantage of the FL algorithm is that when it comes to model aggregation based on averaging methods, the averaging rules do not take into account that the resulting weights associated with different data can have different scales. In this context, the result will end up in favor of the edge model that has the largest scale;
4. Generally speaking, in terms of "statistical heterogeneity", the discussed models follow aggregations of DL networks that share the same architectures. This requires, in case of classification, for example, that the number of output neurons must be the same. This means that we should have the same number of classes in this case. However, in real applications, peripheral devices can be used to monitor a different number of classes with different types depending on the purpose of the monitoring process. At this point, we cannot deny the fact that similar deep network architectures will not handle the actual application of non-IID data in terms of class count.
5. In the context of "privacy concerns", these models have adopted several encryption techniques to protect the privacy of the models and efficiently remedy cyber threats.
6. There is a big issue concerning data selection when validating these learning models. The MNIST dataset, which is generally designed for the purpose of image recognition

and not specifically intended for FL, is widely used in these studies. This is truly a problem that makes results obtained in this work not fully supported by the obtained conclusions in terms of real application. Therefore, to meter the splitting process followed in this case the model will not address issues of FL related to both statistical and system heterogeneity.

Table 1. Recent FL works introduced in the literature.

FL Challenges	Ref	Dataset	Algorithm	Main Contributions in FL Besides Model Aggregation
Expensive communication	[5]	MNIST [6]	FedAvg variants	Using consensus algorithm to make the training of the global model able to adapt to: (i) a heavy communication load and (ii) to avoid communication blocking problems that can be caused by standard FL algorithms such as FedAvg or FSVRG
	[7]	MNIST [6] and comprehensive work of William Shakespeare [8]		Using a bi-directional compression where computationless compression operators are employed to quantify gradients on both global and local model frameworks
	[9]	MNIST [6]		Using a sparse coding algorithm for efficient communication with additive holomorphic encryption including a differential privacy to prevent data leakage
Systems heterogeneity	[10]	MNIST dataset [6], fashion-MNIST dataset [11], and EMNIST dataset [12]		Using an iterative node selection algorithm for efficient management of the FL learning tasks by taking into account the non-synchronization of the delivered messages
	[13]	Heartbeat dataset [14] and the Seizure dataset [15]		Using hierarchical FL in heterogeneous systems by introducing an optimized solution for user assignment and resource allocation by paying attention to variants based on gradient descent while taking unbalanced data into account
	[16]	IMDB [17], Handwritten [18] and ALLAML [19] datasets		Using a semi-supervised FL for heterogeneous transfer learning to take advantage of unlabeled non-overlapping samples and to help reduce overfitting
Statistical heterogeneity	[20]	MNIST [6], Fed-MEx [21], and Fed-Goodreads [22]		Using similarity between clients to model their relationships by involving clients similar gradients to provide better coverage
	[23]	MNIST dataset [6]		Using MHAT for local model problems containing various network architectures while KD techniques are investigated for information extraction and global model update
	[24]	LiveLab dataset		Using a mobile application startup prediction model based on FL
Privacy concerns	[25]	MovieLens [26] and Epinions [27] datasets		Using a trust-based mechanism and extensive reinforcement learning for potential recommender planning and candidate selection.
	[28]	/		Using VANE to perform aggregation without disclosing private information without any interaction between clients
	[29]	MNIST dataset [6]		Using a privacy-preserving FL with DL processes based on the trusted execution environment (TEE)

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

1. Federated networks include a massive number of devices (i.e., millions). This situation definitely slows down communication due to limited resources like bandwidth, energy, and power. As a result, communication methods must iteratively send small messages or model updates as part of the training process, reducing both the number of communications and the size of the messages transmitted each round (see [1], page 52, section "expansive communication"). To overcome the problem of large packets of messages transmission (i.e., expansive communication), the recursive least squares (RLS) method is involved in this work. RLS offers a fast and accurate small scale-based linear programming allowing both the approximation and generalization depending only on a single matrix of weights and not the multiple weighting and nonlinear abstraction processes as with deep networks.

2. RLS depends on the Sherman-Morison-Woodbury (SMW) demonstration to perform anon-iterative model updates requiring very small computational resources compared to gradient decent algorithms.
3. To solve the problem related to weight scales (i.e., statistical heterogeneity), a specific weight scaling and rescaling process has been involved before and after collaborative training.
4. To overcome the problem related to model architecture (i.e., system heterogeneity), a specific mapping process in a sort of feature encoding is designed to unify weights matrix sizes before the collaborative training process.
5. To circumvent the problem of providing more-appropriate data, a specific dataset is generated in this context, addressing all above FL issues in an attempt to provide coherent conclusions.
6. Due to the algorithmic simplicity of the FTR-RLS method and characteristics of the generated dataset, simulations conduction is feasible on ordinary available commercial computers with quad-core microprocessors without any need for higher computing environments.
7. To overcome the problem related to the privacy of the FL model itself (i.e., privacy concerns) FTR-RLS involves a weights encoding process through transpose matrix design as a sort of encryption. This leads to the addition of some difficulty/delay in extracting information from real learning weights through attack experiments.

This paper is organized as follows. Section 2 introduces the generated dataset and the FTR-RLS methods main learning rules. Section 3 provides the results and the discussion of the obtained findings. Section 4 concludes this paper and suggests some future work guidelines.

2. Materials and Methods

This section is dedicated to introducing the dataset and the proposed algorithm used.

2.1. Dataset Generation

Data availability, complexity and drift are the main characteristics on which the reconstruction process of the machine learning model is based [30], especially when it comes to FL, where data from different clients could be completely independent of each other with respect to the above three characteristics of model selection as well as volume, velocity and variety, known as the 3V of data [31]. In this context, we are aware of the existence of a large number of data generated from real scenarios or simulation models [30,32]. However, it should be mentioned that these FL datasets are usually massive with multiple numbers of clients requiring large computation resources, GPUs, and CPUs with multiple cores limiting hence its exploration. From this perspective, our goal is to provide a standard dataset that can be used to test FL algorithms for both small-scale machine learning and DL in addition to addressing FL challenges in a real application. Therefore, one of our primary concerns when generating data is to ensure that data analysis and model reconstruction can be performed on available commercial computers. The dataset is therefore designed to be applied to four clients, which means that it will scale well with quad-core microprocessors.

In this work, the non-IID is collected as a 2D numerical feature space generated for classification purposes through the data engine provided in [33]. Thus, samples are delivered as pseudo-parallel lines mainly generated according to some specific parameters of normal distribution, which have been introduced in Table 2. Since our primary goal is to provide a non-IID that can address decentralized federated learning challenges as in real applications (i.e., expensive communication, systems heterogeneity, statistical heterogeneity, and privacy concerns), the following tasks are considered.

1. To emulate real conditions of data complexity related to statistical heterogeneity, d_x and d_y are used as the main parameters to generate a different version of data complexity levels in terms of cardinality. This process was also adopted to make

sure that data will be applicable for both conventional small-scale machine learning algorithms and DL.

2. To respond to a real problem of systems heterogeneity, the generated dataset contains four levels of complexity: S01, S02, S03 and S04. At each level, four subsets are generated independently for four particular clients, C01, C02, C03 and C04. The subsets generated for each client are independent of each other in terms of distributions, observation scales, class numbers and class proportions.

Table 2. Main parameter description of dataset generator.

Parameter	Description
μ_θ	Mean of the radians of clusters which are originally drawn from a normal distribution
δ_θ	Standard deviation of cluster angle
l_c	Number of clusters
d_x	Average distance between classes centers within X axis
d_y	Average distance between classes centers within Y axis
μ_l	Mean of length of which originally are drawn from normal distribution
δ_l	Standard deviation of clusters lengths
δ_f	Clusters fatness which is the standard deviation of the distance between each point and the projection line
N	Number of observations in the generated data

It is worth mentioning that while generating data, only two FL challenges are considered, i.e., statistical heterogeneity and systems heterogeneity, while the other two challenges, i.e., expensive communication and privacy concerns, should be resolved during the model construction process. Additionally, the random seed engine has been given default settings to ensure that the data versions in terms of complexity share some information about the feature space, as shown in Table 3. Table 3 is dedicated to the presentation of the main characteristics of the dataset.

Table 3. Tuned parameters for dataset generation.

Client→	**C01**									**C02**								
Subset↓	μ_θ	δ_θ	l_c	d_x	d_y	μ_l	δ_l	δ_f	N	μ_θ	δ_θ	l_c	d_x	d_y	μ_l	δ_l	δ_f	N
S01	$\pi/2$	$\pi/2$	2	30	30	2	2	10	1000	$\pi/2$	$\pi/2$	3	30	30	2	2	10	1000
S02	$\pi/2$	$\pi/2$	2	20	20	2	2	10	1000	$\pi/2$	$\pi/2$	3	20	20	2	2	10	1000
S03	$\pi/2$	$\pi/2$	2	10	10	2	2	10	1000	$\pi/2$	$\pi/2$	3	10	10	2	2	10	1000
S04	$\pi/2$	$\pi/2$	2	0	0	2	2	10	1000	$\pi/2$	$\pi/2$	3	0	0	2	2	10	1000
Client→	**C03**									**C04**								
Subset↓	μ_θ	δ_θ	l_c	d_x	d_y	μ_l	δ_l	δ_f	N	μ_θ	δ_θ	l_c	d_x	d_y	μ_l	δ_l	δ_f	N
S01	$\pi/2$	$\pi/2$	4	30	30	2	2	10	1000	$\pi/2$	$\pi/2$	5	30	30	2	2	10	1000
S02	$\pi/2$	$\pi/2$	4	20	20	2	2	10	1000	$\pi/2$	$\pi/2$	5	20	20	2	2	10	1000
S03	$\pi/2$	$\pi/2$	4	10	10	2	2	10	1000	$\pi/2$	$\pi/2$	5	10	10	2	2	10	1000
S04	$\pi/2$	$\pi/2$	4	0	0	2	2	10	1000	$\pi/2$	$\pi/2$	5	0	0	2	2	10	1000

For better illustration of data characteristics, including complexity levels (i.e., S01–S04), subsets of clients (i.e., C01–C04), number of classes (i.e., 2–5) and their distributions, data visualization through scatter plots is given in Figure 1. It can be seen that centers of data classes have different coordinates and different sparsity levels for each client where all samples are in the range [0, 1]. Moreover, at each complexity level, we perceive close distances resulting in the higher cardinality at each level.

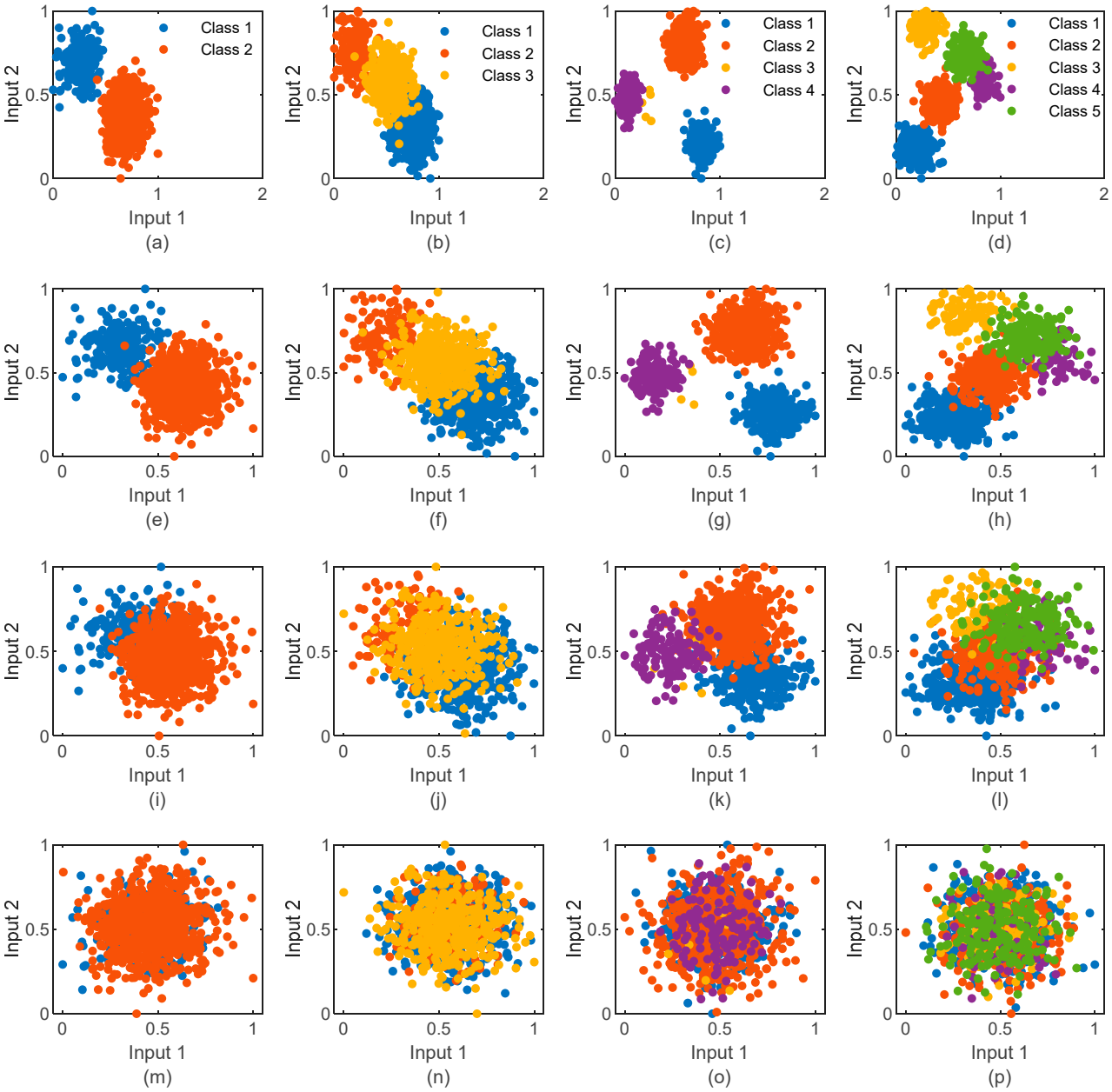

Figure 1. Illustration of dataset scatters: (**a**–**d**) Data scatters of S01 for C01,C02, C03, C04; (**e**–**h**) Data scatters of S02 for C01,C02, C03, C04; (**i**–**l**) Data scatters of S03 for C01,C02, C03, C04; (**m**–**p**) Data scatters of S04 for C01,C02, C03, C04.

2.2. Federated Transfer Learning Recursive Least Squares

FTL-RLS rules are inspired by basic RLS methods [34]. Accordingly, this subsection is devoted to the introduction of the FTL-RLS passing through ordinary LS. RLS typically processes data that arrive in sequences over time as model updates are needed. An RLS model could be trained in two main phases, namely the initial phase and the recursive learning phase [34]. In the initial phase, the model requires a feature map of an initial mini-batch of input data x_m and corresponding targets y_m, where m represents the index of the training mini-batch ($m \leftarrow 1$ in the initial phase). As a result, training weights of RLS will be determined using the covariance matrix of feature maps P_m as in (1), the inverse of $\varphi\ (x_m)$, and y_m as in (2).

$$P_m = \left[\varphi(x_m) \cdot \varphi(x_m)^T\right]^{-1} \tag{1}$$

$$W_m = P_m \cdot \varphi(x_m)^{-1} \cdot y_m \tag{2}$$

In the sequential phase, while $m \leftarrow m+1 : m_{max}$ the model update will be done using SMW formula which leads to equations (4) to (6). K is a gain matrix and ξ is the prediction error.

$$K_{m+1} = P_m \cdot \varphi(x_{m+1}) / (\varphi(x_{m+1})^T \cdot P_m \cdot \varphi(x_{m+1})^T) \tag{3}$$

$$P_{m+1} = P_m \cdot \varphi(x_{m+1})^T P_m \tag{4}$$

$$\xi_{m+1} = y_{m+1} - \varphi(x_{m+1})^T W_m \tag{5}$$

$$W_{m+1} = W_{m+1} - P_{m+1} \varphi(x_{m+1})^T \xi_{m+1} \tag{6}$$

RLS can be simply presented as in Algorithm 1.

Algorithm 1. RLS algorithm.

Inputs: $x_m, x_{m+1}, y_m, y_{m+1}, m, m_{max}$
Outputs: $W_{m_{max}}$
% *Initialization*: $m \leftarrow 1$
$P_m = [\varphi(x_m) \cdot \varphi(x_m)^T]^{-1}$;
$W_m = P_m \cdot \varphi(x_m)^{-1} \cdot y_m$;
% *Recursive learning phase*
For $m \leftarrow 2 : m_{\max}$
$K_{m+1} = P_m \cdot \varphi(x_{m+1}) / (\varphi(x_{m+1})^T \cdot P_m \cdot \varphi(x_{m+1})^T)$;
$P_{m+1} = P_m \cdot \varphi(x_{m+1})^T P_m$;
$\xi_{m+1} = y_{m+1} - \varphi(x_{m+1})^T W_m$;
$W_{m+1} = W_{m+1} - P_{m+1} \varphi(x_{m+1})^T \xi_{m+1}$;
End (For)

FTL-RLS, which is presented in the flow diagram of Figure 2, follows a similar weight initialization mechanism. However, this time the model is designed for each client $i = 1 : n$ for several FL rounds round $k \leftarrow 1 : k_{max}$ when sent to FL updates center.

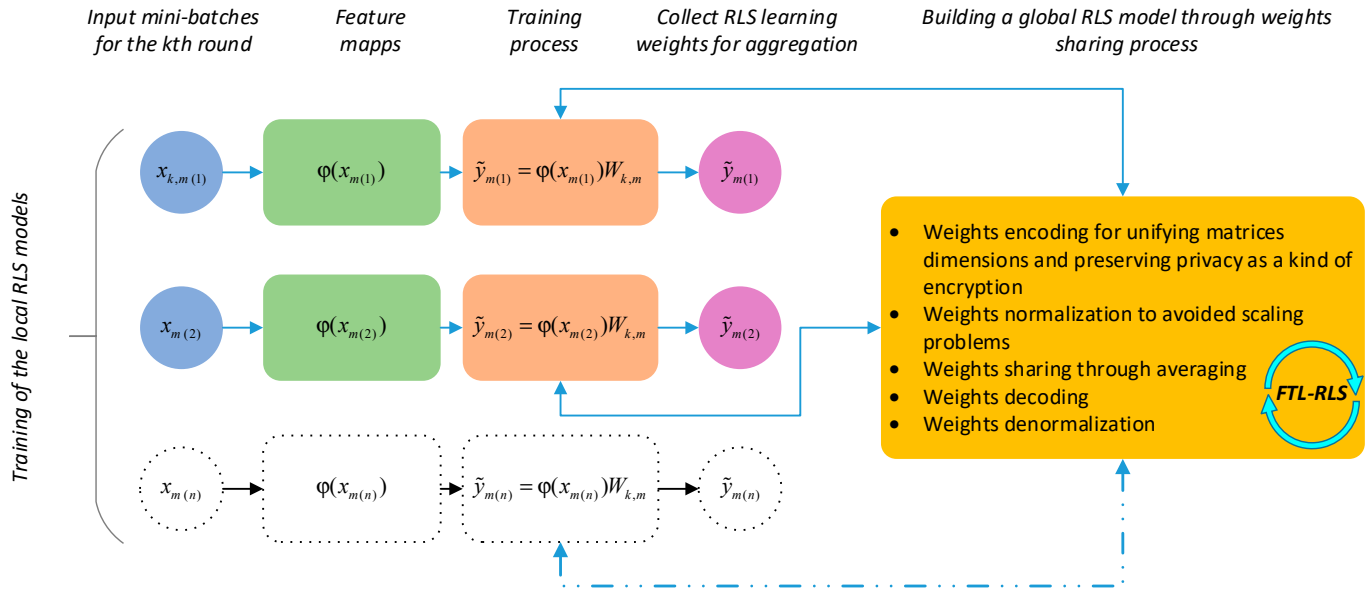

Figure 2. Flow diagram of the proposed FTL-RLS.

The weight sharing process described by the last block of Figure 2 dictates the necessary steps for improving the generalization of the model without data sharing (keeping data private). This means that only local model training parameters will be collected in a distributed parallel computing pool and merged to achieve better performance through transfer learning between different models.

In this context, the initialization phase of FTR-RLS is computed as illustrated by (7) and (8) for $k = 1$.

$$P_{k,m(i)} = \left[\varphi_i(x_m) \cdot \varphi_i(x_m)^T\right]^{-1} \tag{7}$$

$$W_{k,m(i)} = P_{k,m(i)} \cdot \varphi_i(x_m)^{-1} \cdot y_{m(i)} \tag{8}$$

After that, the recursive learning phase for the same round, clients, and coming mini-batches can be done using Equations (9) to (12).

$$K_{k,m+1(i)} = P_{k,m(i)} \cdot \varphi_i(x_{m+1}) / (\varphi_i(x_{m+1})^T \cdot P_{k,m(i)} \cdot \varphi_i(x_{m+1})^T) \tag{9}$$

$$P_{k,m+1(i)} = P_{k,m(i)} \cdot \varphi_i(x_{m+1})^T P_{k,m(i)} \tag{10}$$

$$\xi_{k,m+1(i)} = y_{k,m+1(i)} - \varphi_i(x_{m+1})^T W_{k,m(i)} \tag{11}$$

$$W_{k,m+1(i)} = W_{k,m+1(i)} - P_{k,m+1(i)} \varphi_i(x_{m+1})^T \xi_{k,m+1(i)} \tag{12}$$

After several model updates, the model will be required to send to a further central training process without data sharing for different rounds $k \leftarrow k + 1 : k_{max}$. Thus, our proposition to model updates will be as follows. Normally, due to the different number of features in data delivered for each client, the learning weights of the FTL-RLS will be different in terms of matrix size. In this case, we need to follow certain procedures in order to make sure that they are transformed into the global updates center with a unique size. Accordingly, we should make sure that the feature maps hold the same mapping parameters which at least will lead to unify one side of the weight matrices. Second, an encoding based transpose such as the weights matrices in (13) will make all of them share the same sizes. C is a superscript referring to encoded weights.

$$W^C_{k,m+1(i)} = W_{k,m+1(i)} \cdot W^T_{k,m+1(i)} \tag{13}$$

After that, another problem of non-IID related to weights scales could occur in this case, yielding the training process to end up in favor of the weights with large scales. For this reason, our contribution in this context is to normalize the learning weights using the standard deviation $\delta_{k,m+1(i)}$ and the mean values $\mu_{k,m+1(i)}$ of each matrix of weights as in (14). N is a superscript referring to normalized encoded weights.

$$W^N_{k,m+1(i)} = \frac{W^C_{k,m+1(i)} - \mu_{k,m+1(i)}}{\delta_{k,m+1(i)}} \tag{14}$$

when weights aggregation is done at the central server as addressed in (15), a denormalization and decoding processes is required as provided in (16) and (17), respectively. It should be mentioned that in our work we consider that the inverse of a matrix indicated by superscript (-1) is defined as the pseudo inverse method as default.

$$W_{G,k} = \sum_{i=1}^{n} W^N_{k,m+1(i)} \tag{15}$$

$$W^{Dn}_{k,m+1(i)} = \frac{W_G \cdot \delta_{k(i)}}{\mu_{k(i)}} \tag{16}$$

$$W_{k,m+1(i)} = W^{Dc}_{k,m+1(i)} \cdot \left(W^T_{k,m(i)}\right)^{-1} \tag{17}$$

To make sure that FTR-RLS rules are clearly presented, Algorithm 2 is introduced to elucidate its main steps as well the main difference between FTL-RLS and ordinary RLS.

Algorithm 2. FTL-RLS algorithm.

Inputs: $x_m, x_{m+1}, y_m, y_{m+1}$, m, n, k_{max}, m_{max}
Outputs: $W_{k,m_{max}(i)}$
For $k = 1:\ k_{max}$
% Initialization: $m \leftarrow 1$
% Activate parallel computing pool for $i = [1:n]$ *clients for local training*
If $k \leftarrow 1$
$P_{k,m(i)} = [\varphi_i(x_m) \cdot \varphi_i(x_m)^T]^{-1}$;
$W_{k,m(i)} = P_{k,m(i)} \cdot \varphi_i(x_m)^{-1} \cdot y_m$;
End (If)
% Recursive learning phase
For $m \leftarrow 2 : m_{max}$
$K_{k,m+1(i)} = P_{k,m(i)} \cdot \varphi_i(x_{m+1}) / (\varphi_i(x_{m+1})^T \cdot P_{k,m(i)} \cdot \varphi_i(x_{m+1})^T)$;
$P_{k,m+1(i)} = P_{k,m(i)} \cdot \varphi_i(x_{m+1})^T P_{k,m(i)}$;
$\xi_{k,m+1(i)} = y_{k,m+1(i)} - \varphi_i(x_{m+1})^T W_{k,m(i)}$;
$W_{k,m+1(i)} = W_{k,m+1(i)} - P_{k,m+1(i)} \varphi_i(x_{m+1})^T \xi_{k,m+1(i)}$;
% Encoding with the transpose
$W^C_{k,m+1(i)} = W_{k,m+1(i)} \cdot W^T_{k,m+1(i)}$;
End (For)
% Start federated training process
% Normalization of weights matrices
$W^N_{k,m+1(i)} = \frac{W^C_{k,m+1(i)} - \mu_{k,m+1(i)}}{\delta_{k,m+1(i)}}$;
% Deactivate parallel computing pool and work on normal CPU mode
% Initiate collaborative training(aggregation)
$W_{G,k} = \sum_{i=1}^{n} W^N_{k,m+1(i)}$;
% Activate parallel computing pool for $i = [1:n]$ *clients weights processing*
% Demoralization of weights matrices
$W^{Dn}_{k,m+1(i)} = \frac{W_G \cdot \delta_{k(i)}}{\mu_{k(i)}}$;
% Decode with inverse of the transpose and model update weights
$W_{k,m+1(i)} = W^{Dc}_{k,m+1(i)} \cdot (W^T_{k,m(i)})^{-1}$;
% Deactivate parallel computing pool and work on CPU mode
End (For)

3. Results and Discussion

During our experiments, the evaluation of FTL-RLS is carried out under well-defined circumstances. Computational resources involve a personal computer (PC) with an i7 microprocessor, 16 GB of RAM, and a 12 MB cache of RAM, while MATLAB r2018b was involved in parallel computing simulations. The data have been divided into training and testing sets with 80% and 20% ratios, respectively. For the training set, data have been divided into equal sized mini-batches of 10 instances, except for the initialization process of RLS, which generally requires a considerable amount of training samples (100 samples are used). After 10 training updates (10 mini-batches), local models are sent to the server for the collaborative training of weights without training data. This is reflected by $m = \left(\left(\frac{80\times1000}{100} - 100\right)/10\right) + 1) = 71$ mini-batches, and $k = \frac{71}{10} \approx 7$ rounds. For categorical variables related to targets, one-hot-key known as dummy encoding is involved in this case. This means that the size of training weights will be different for each client depending on the number of classes. For feature maps $\varphi_i(x_m)$ of FTL-RLS, a linear mapping with a (2×100) full rank matrix Ω followed by a logistic function transformation is adopted in this work, as explained in (18).

$$\varphi_i(x_m) = \frac{1}{1 + e^{\Omega x_m}} \tag{18}$$

Accordingly, in this section we will showcase performances of FTL-RLS at different levels of complexity for different clients. Therefore, first we will start by explaining the effects of the proposed weights encoding/decoding process besides normalization/renormalization while we will show some visual explanation of the accuracy of the method. Second, we will depict the behavior of the training model towards each stage

of complexity in terms of classification accuracy. It should be mentioned that the main objective of this study is to analyze and investigate the behavior of the RLS method in FTL. Consequently, the experimental analysis is also directed towards this point. This means that this work will not engage in comparisons with state of the art literature papers as the model did not consider any improvements in context of regularization, adaptive learning, feature maps optimization, hyperparameters selection, feature selections, and training data splitting, etc. However, the main conclusion will surround assessing the accuracy and speed of the learning model under this experiment compared to well-known architectures of DL such as long-short term memory (LSTM), convolutional neural networks (CNN), and a shallow artificial neural network (ANN) that adopts the FedAvg training philosophy. The reason for this comparison is to shed light on the accuracy of FTL-RLS and the training speed features. It should be mentioned that these algorithms also adopt the same weight transportation methodology due to the non-IID nature of the data, especially when the one-hot-keys of each neural network are different.

For weight processing, a well-prepared example will be illustrated in this case. We have used a 100×5 matrix of learning weights that resulted from the analytical relationship between feature maps $\varphi_4(x_{10})$ and targets $\widetilde{y}_{10(4)}$ of subset S01-C04 to expose very important illustrations related to reconstruction performances. Figure 3 is therefore elucidated to show weighs trip for local model to global mode and afterwards. The heat map in Figure 3a shows that the original learning weights were red spots indicate most important learning parameters directing learning process. The learning weights are first decoded using a transpose as in Figure 3b to make sure that FTL is achievable in terms of weight dimensions. After that, these weights are scaled using mean value and standard deviation to make sure that the aggregation process will not be biased. The color bar values in Figure 3c point out how the values of Figure 3b are scaled thanks to the normalization process. Figure 3d showcases the effect of the aggregation process on learning weights. At this stage, no more details can be revealed by visual observation of the heat map. Instead, root means squared error (RMSE) of the reconstruction will judge the quality of process encoding/and normalization/denormalization. The weights scale is restored after aggregation based on previous information of local training, while the decoding process is achieved with the help of transpose inverse of the old weights matrix. Accordingly, reconstruction RMSE $= 7.1118 \times 10^{-7}$ dictates that training weights have an acceptable quality and not distorted learned model. To sum-up, the aggregation method of FTL-RLS proved its ability to keep information regarding learning weights, even under the aggregation process.

When training the FTL-RLS model on sequentially delivered mini-batches, the classification accuracy of training has been recorded at each model update, including local and global ones. Accordingly, Figure 4 is dedicated to introducing the collected results. From Figure 4, the following conclusions can be drawn:

1. FTR-RLS performs well at S01 for the different clients, especially C01 and C02 when data classes are only two and three, respectively. When the number of classes gets bigger the training process becomes difficult;
2. For the other subsets of S02,S03, and S04, the more the complex the data are, the weaker the training becomes. In this context, we are noticing deterioration in FTR-RLS in terms of convergence speed and accuracy. This means that the model is getting slower when reaching its optimal accuracy, which is also reduced due to data complexity and the increased number of classes;
3. By the increase of data complexity, we observe more fluctuation in curves of classification accuracy behavior as a sort of noise. This fluctuation reflects the instability of the training process as a reaction to higher levels of cardinality.
4. In Figure 4a–c, we observe that there is a distance between the performances curves of (C01, C02, C03) and (C04). This distance is getting closer during complexity until it reaches Figure 4c, where the training process performances for all clients are roughly becoming the same due to complexity.

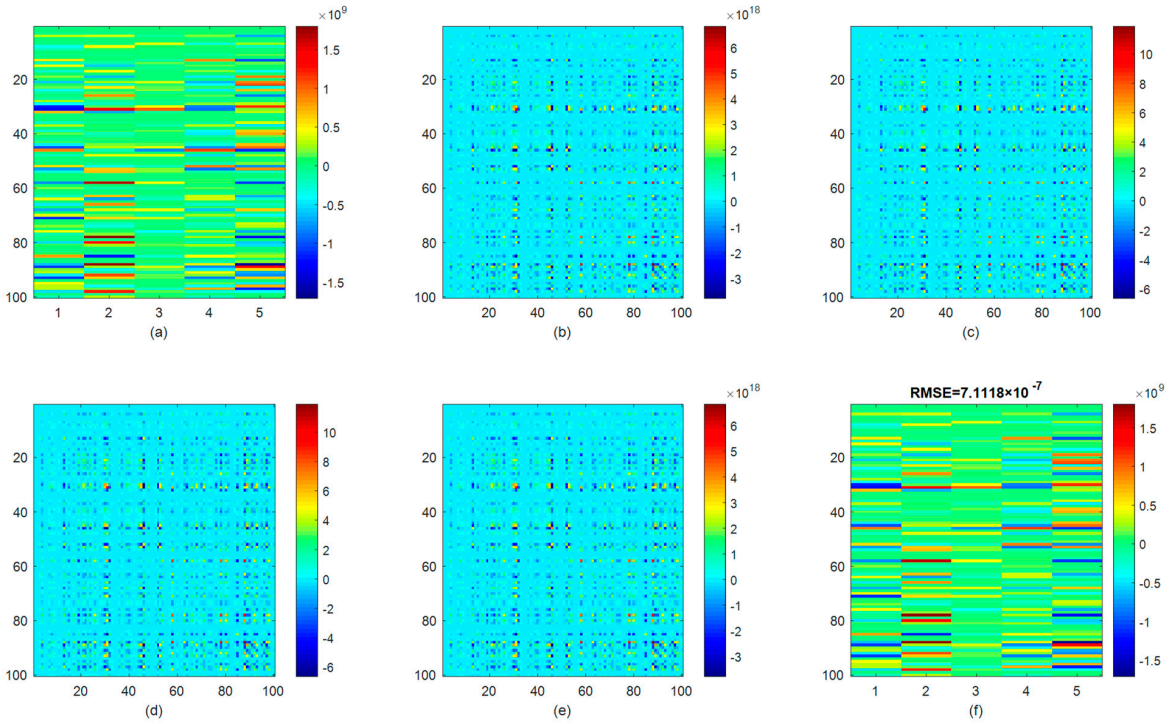

Figure 3. Illustrations of the effects of the proposed weights encoding/decoding and the normalization/denormalization method on training weights: (**a**) Analytically determined weights with the RLS method $W_{1,10(4)}$; (**b**) encoded weights with transpose matrix for unifying dimensions with other clients models $W^{C}_{1,10(4)} = W_{1,10(4)} \cdot W^{T}_{1,10(4)}$; (**c**) Normalized version of encoded weights $W^{N}_{1,10(4)}$; (**d**) Weights of global model after aggregation $W_{G,k}$; (**e**) Denormalized version of training weights after aggregation $W^{Dn}_{1,10(4)}$; (**f**) Decoded weights after aggregation and denormalization $W_{1,10(4)}$.

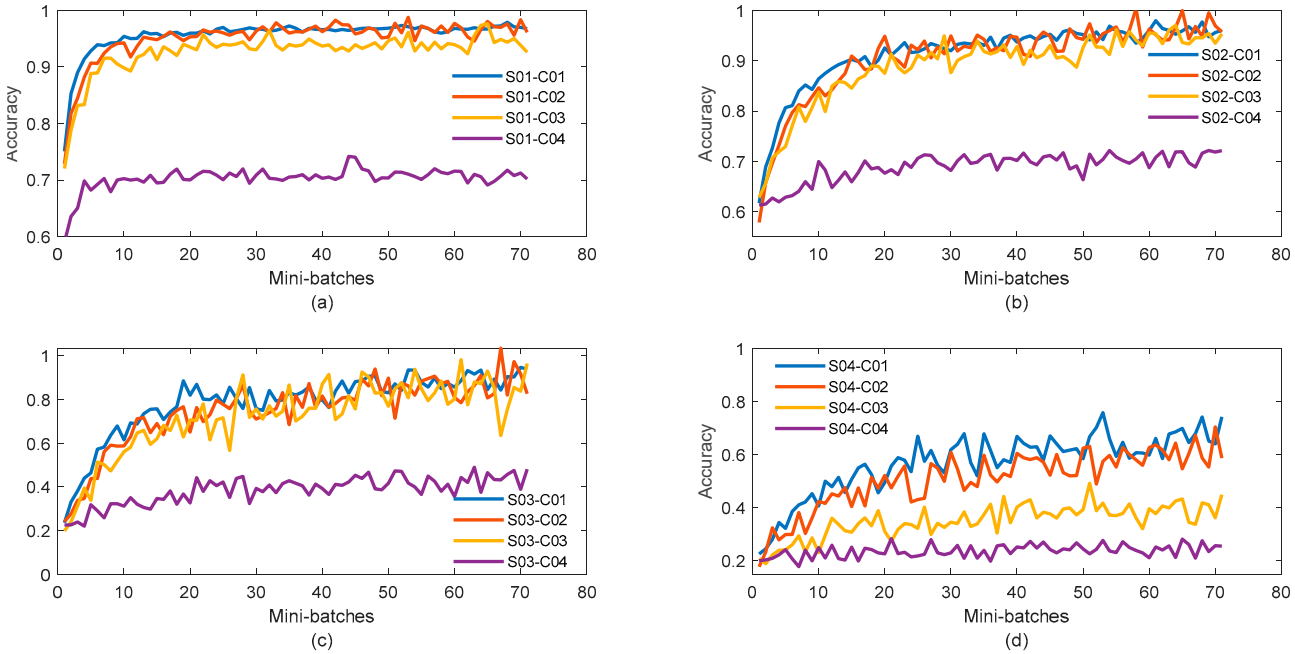

Figure 4. FTL-RLS classification performances: (**a**–**d**) Classification accuracy variation for subset S01, S02, S03, and S04 respectively.

Generally speaking, FTR-RLS has good performances under data with less complexity. These performances are reflected by higher convergence speed, less fluctuation which

mean stability, and higher accuracy. However, these features are degraded under increased data complexity.

Besides training performances assessed by the classification accuracy of the training set, classification accuracy of the test set of the global model at the final mini-batch of the final FL round is also used as a measure to assess the capability of FTL-LRS in classifying new unseen samples to the model. As introduced in Table 4, we can see that the LSTM network and CNN also are involved. Furthermore, comparison criteria were focused on classification accuracy for each local data independently as well as for the entire data in general represented by the mean value. For the three other adopted algorithms (i.e., ANN, LSTM, and CNN), the feature maps are adjusted to be as similar as possible to the number of parameters in the features maps $\varphi_i(x_m)$ of the FTR-LRS algorithm. In this context, both ANN and LSTM use a single hidden layer with 100 neurons. Concerning CNN, we employed a single feature mapping process in which a one-dimensional convolutional layer and a pooling layer where local receptive fields are tuned to give a close number to parameters in features maps $\varphi_i(x_m)$ after the pooling process. In addition, the hyperparameters of the three models including regularization, learning rate, activation functions, etc., are given by the system through a grid search mechanism except for FL parameters, which are adjusted the same way as FTL-RLS. All of this architecture and hyperparameters tuning process are identified to make sure of a fair comparison as close as possible between these algorithms.

Table 4. Comparison of FL algorithms performance.

Algorithm	Subset	Accuracy (x 100%)				Shared Parameters Sizes (Bytes)	Training Time (Seconds)
		C01	C02	C03	C04		
FTR-RLS	**S01**	0.978	0.993	0.985	0.731	Size of encoded weights $W^{C}_{k,m(i)}$ $\approx 80,000$	$\approx$ **2.43**
	S02	0.973	0.954	0.978	0.725		
	S03	0.978	0.960	0.932	0.504		
	S04	0.764	0.486	0.582	0.320		
	Mean	0.8027					
ANN (FedAvg)	**S01**	0.977	0.993	0.970	0.734	Weights and biases of inputs, hidden layers and output layers: $\approx 160,846$	$\approx$45.63
	S02	0.977	0.958	0.980	0.728		
	S03	0.972	0.954	0.936	0.500		
	S04	0.711	0.452	0.542	0.298		
	Mean	0.7926					
LSTM (FedAvg)	**S01**	0.999	0.997	0.992	0.730	Weights and biases of inputs, hidden layers and output layers: $\approx 160,846$	$\approx$108.22
	S02	0.995	0.979	0.993	0.721		
	S03	0.999	0.975	0.999	0.741		
	S04	0.773	0.428	0.585	0.329		
	Mean	0.8272					
CNN (FedAvg)	**S01**	1.000	1.000	0.995	0.732	Weights and biases of local fields, convolutional layers, pooling layers, and output layers, etc.: $> 160,846$	$\approx$122.36
	S02	1.050	1.034	1.048	0.761		
	S03	0.997	0.973	0.997	0.740		
	S04	0.842	0.466	0.637	0.358		
	Mean	0.8519					

By referring to Table 4, the following conclusions can be drawn:

1. For each algorithm in general classification accuracies express significant decrease in performances when moving from client C01 to C02. This explains that the number of classes affects the predictions of training models.
2. For each algorithm, moving from different levels of data complexity resembled by distance between classes also has a significant impact in performances reduction of performances of learning algorithm.

3. The different subsets of C04's low classification results explain that it is the most challenging part in the generated data.
4. It is undeniable that CNN and LSTM could achieve more classification accuracy which is in this case better than FTL-RLS. However, we cannot deny that blackbox models used in this case are strengthened by many default features such as regularization, parameter selection through grid search, adaptive learning optimizers, etc., while FTL-RLS has only basic rules of learning. In this context, the performance of FTL-RLS compared to these algorithms is comparable, which explains its strength in keeping classification performances under FL.
5. CNN achieves better results than LSTM owing to its known capabilities in pattern separation through mapping-based local receptive fields.
6. ANN and RLS provide almost the same results in this case.
7. One of the important FTL-RLS features is "computational time" and "algorithmic simplicity", related to both "algorithmic architecture" itself (number of instructions) and the "non-iterative" nature of the RLS algorithm. In this context, the training time of FTL-RLS (i.e., 2.43 s) clarifies that the algorithm performances are way too far better and the algorithmic architecture can be easily constructed and implemented in low computational costly hardware unlike ANN, LSTM and CNN.
8. Shared parameter sizes (i.e., weights, biases, etc.) of learning algorithms show that the FTL-RLS encoding process reduces messages sizes with more than 50% compared to ANN, LSTM, CNN and fair comparison.

4. Conclusions

This paper has introduced a new federated learning algorithm whose primary target is commination expansiveness among other FL challenges. The learning algorithm is inspired from basic learning rules of the RLS algorithm. The algorithm was applied to a specific generated data that emulates real FL challenges and was specifically designed to fit quad-core microprocessors available in commercial computers. The reason for designing such a dataset is to make sure that researchers are able to replicate experiments and discover more about the findings of this work. The application of this designed FTL-RLS algorithm led to many conclusions, which can be summed up in both training speed and accuracy of the model compared to existing complicated architectures trained based on gradient descent algorithms. It also contributes to communication expensive reduction, thereby leading to lower computational costs. It should be mentioned that this paper studies FTL-RLS with no aforementioned additive features of learning rules optimization or hyperparameter selection. Therefore, it would be better if future works concentrated on this axis and provided more insights into the algorithm using not only the same dataset but also other real datasets for more generalization.

Author Contributions: Conceptualization, T.B. (Tarek Berghout); methodology, T.B. (Tarek Berghout); software, T.B. (Tarek Berghout); validation, T.B. (Tarek Berghout), M.A.F. and M.B.; formal analysis, T.B. (Tarek Berghout); investigation, T.B. (Tarek Berghout); resources, T.B. (Tarek Berghout); data curation, T.B. (Tarek Berghout); writing—original draft preparation, T.B. (Tarek Berghout); writing—review and editing T.B. (Tarek Berghout), T.B. (Toufik Bentrcia), M.A.F. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Yang, Q.; Liu, Y.; Cheng, Y.; Kang, Y.; Chen, T.; Yu, H. Federated Learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2019**, *13*, 1–207. [CrossRef]
2. Li, L.; Fan, Y.; Tse, M.; Lin, K.-Y. A review of applications in federated learning. *Comput. Ind. Eng.* **2020**, *149*, 106854. [CrossRef]
3. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
4. Nilsson, A.; Smith, S.; Ulm, G.; Gustavsson, E.; Jirstrand, M. A performance evaluation of federated learning algorithms. *DIDL 2018-Proc. 2nd Work. Distrib. Infrastructures Deep Learn. Part Middlew.* **2018**, *2018*, 3286559. [CrossRef]
5. Liu, B.; Ding, Z. A consensus-based decentralized training algorithm for deep neural networks with communication compression. *Neurocomputing* **2021**, *440*, 287–296. [CrossRef]
6. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
7. Fang, C.; Guo, Y.; Hu, Y.; Ma, B.; Feng, L.; Yin, A. Privacy-preserving and communication-efficient federated learning in Internet of Things. *Comput. Secur.* **2021**, *103*, 102199. [CrossRef]
8. Shakespeare, W. The Complete Works of William Shakespeare. 2020. Available online: http://www.gutenberg.org/ebooks/100 (accessed on 3 September 2022).
9. Asad, M.; Moustafa, A.; Ito, T. FedOpt: Towards communication efficiency and privacy preservation in federated learning. *Appl. Sci.* **2020**, *10*, 2864. [CrossRef]
10. Chen, Z.; Liao, W.; Hua, K.; Lu, C.; Yu, W. Towards asynchronous federated learning for heterogeneous edge-powered internet of things. *Digit. Commun. Netw.* **2021**, *7*, 317–326. [CrossRef]
11. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
12. Cohen, G.; Afshar, S.; Tapson, J.; van Schaik, A. EMNIST: An extension of MNIST to handwritten letters. *arXiv* **2017**, arXiv:1702.05373.
13. Abdellatif, A.A.; Mhaisen, N.; Mohamed, A.; Erbad, A.; Guizani, M.; Dawy, Z.; Nasreddine, W. Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data. *Future Gener. Comput. Syst.* **2022**, *128*, 406–419. [CrossRef]
14. ECG. Heartbeat Categorization Dataset: Segmented and Preprocessed ECG Signals for Heartbeat Classification. 2018. Available online: https://www.kaggle.com/datasets/shayanfazeli/heartbeat (accessed on 28 July 2022).
15. Schomer, D.L.; Da Silva, F.L. *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*; Lippincott Williams & Wilkins: Philadelphia, PA, USA, 2012.
16. Feng, S.; Li, B.; Yu, H.; Liu, Y.; Fellow, Q.Y. Semi-Supervised Federated Heterogeneous Transfer Learning. *Knowledge-Based Syst.* **2015**, *14*, 109384. [CrossRef]
17. IMDB Dataset. Available online: https://drive.google.com/file/d/0B8yp1gOBCztyN0JaMDVoeXhHWm8/edit?resourcekey=0-y9_nzlfIi3jTOoMJ0xzahw (accessed on 28 July 2022).
18. Van Breukelen, M.; Duin, R.P.W.; Tax, D.M.J.; Den Hartog, J.E. Handwritten digit recognition by combined classifiers. *Kybernetika* **1998**, *34*, 381–386.
19. ALLAML Dataset. Available online: https://jundongl.github.io/scikit-feature/datasets.html (accessed on 28 July 2022).
20. Palihawadana, C.; Wiratunga, N.; Wijekoon, A.; Kalutarage, H. FedSim: Similarity guided model aggregation for Federated Learning. *Neurocomputing* **2022**, *483*, 432–445. [CrossRef]
21. Wijekoon, A.; Wiratunga, N.; Cooper, K.; Bach, K. Learning to recognise exercises in the self-management of low back pain. In Proceedings of the Thirty-Third International Flairs Conference, Miami Beach, PA, USA, 7–20 May 2020.
22. Goodreads Datasets. Available online: https://sites.google.com/eng.ucsd.edu/ucsdbookgraph (accessed on 30 July 2022).
23. Hu, L.; Yan, H.; Li, L.; Pan, Z.; Liu, X.; Zhang, Z. MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. *Inf. Sci.* **2021**, *560*, 493–503. [CrossRef]
24. Li, S.; Lv, L.; Li, X.; Ding, Z. Mobile app start-up prediction based on federated learning and attributed heterogeneous network embedding. *Future Internet* **2021**, *13*, 256. [CrossRef]
25. Wahab, O.A.; Rjoub, G.; Bentahar, J.; Cohen, R. Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems. *Inf. Sci.* **2022**, *601*, 189–206. [CrossRef]
26. MovieLens. Available online: https://grouplens.org/datasets/movielens/1m (accessed on 30 July 2022).
27. Epinions. Available online: http://www.epinions.com (accessed on 30 July 2022).
28. Wang, F.; Zhu, H.; Lu, R.; Zheng, Y.; Li, H. A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent. *Inf. Sci.* **2021**, *552*, 183–200. [CrossRef]
29. Chen, Y.; Luo, F.; Li, T.; Xiang, T.; Liu, Z.; Li, J. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Inf. Sci.* **2020**, *522*, 69–79. [CrossRef]
30. Berghout, T.; Benbouzid, M.; Muyeen, S.M. Machine Learning for Cybersecurity in Smart Grids: A Comprehensive Review-based Study on Methods, Solutions, and Prospects. *Int. J. Crit. Infrastruct. Prot.* **2022**, *38*, 100547. [CrossRef]
31. Berghout, T.; Benbouzid, M. A Systematic Guide for Predicting Remaining Useful Life with Machine Learning. *Electronics* **2022**, *11*, 1125. [CrossRef]

32. Berghout, T.; Benbouzid, M. EL-NAHL: Exploring labels autoencoding in augmented hidden layers of feedforward neural networks for cybersecurity in smart grids. *Reliab. Eng. Syst. Saf.* **2022**, *226*, 108680. [CrossRef]
33. Fachada, N.; Rosa, A.C. generateData—A 2D data generator. *Softw. Impacts* **2020**, *4*, 100017. [CrossRef]
34. Berghout, T.; Mouss, L.; Kadri, O.; Saïdi, L.; Benbouzid, M. Aircraft Engines Remaining Useful Life Prediction with an Improved Online Sequential Extreme Learning Machine. *Appl. Sci.* **2020**, *10*, 1062. [CrossRef]