Federated Reservoir Computing Neural Networks

Davide Bacciu
Dept. of Computer Science
University of Pisa
Pisa, Italy
bacciu@di.unipi.it

Daniele Di Sarli
Dept. of Computer Science
University of Pisa
Pisa, Italy
daniele.disarli@phd.unipi.it

Pouria Faraji
Dept. of Computer Science
University of Pisa
Pisa, Italy
p.faraji@studenti.unipi.it

Claudio Gallicchio
Dept. of Computer Science
University of Pisa
Pisa, Italy
gallicch@di.unipi.it

Alessio Micheli
Dept. of Computer Science
University of Pisa
Pisa, Italy
micheli@di.unipi.it

***Abstract*—A critical aspect in Federated Learning is the aggregation strategy for the combination of multiple models, trained on the edge, into a single model that incorporates all the knowledge in the federation. Common Federated Learning approaches for Recurrent Neural Networks (RNNs) do not provide guarantees on the predictive performance of the aggregated model. In this paper we show how the use of Echo State Networks (ESNs), which are efficient state-of-the-art RNN models for time-series processing, enables a form of federation that is optimal in the sense that it produces models mathematically equivalent to the corresponding centralized model. Furthermore, the proposed method is compliant with privacy constraints. The proposed method, which we denote as Incremental Federated Learning, is experimentally evaluated against an averaging strategy on two datasets for human state and activity recognition.**

I. Introduction

In a centralized setting, a Machine Learning algorithm can make use of all the available training data to produce a predictive model that best generalizes to unseen data. Unfortunately, a centralized setting is not always feasible. When the data comes from multiple independent devices, constraints such as *network connectivity*, *bandwidth*, and *privacy preservation* can make it impossible to aggregate the training data within a centralized location.

In a typical Federated Learning scenario [1], the aforementioned problem is tackled by letting each client produce a local Machine Learning (ML) model trained on just the locally available data. Then, instead of the raw data, it is the models that are transferred to a centralized location such as a server. In the server, the models must be aggregated by some kind of strategy (e.g., averaging the weights) and then sent back to the clients if they need it for inference or further training. The critical point for an effective federation lies in the aggregation strategy, which ideally should produce a single compact model that incorporates all the knowledge from each client. However, due to the notorious difficulty in the interpretation of the weights of a neural network, it is not easy to give guarantees about the outcome of the aggregation.

Due to the relevance of Federated Learning in the case of clients collecting sensor data, in this paper we focus on federation techniques for Recurrent Neural Networks (RNNs), which are ML models especially suited for time-series processing. While there is a vast amount of literature regarding federated RNNs [2]–[6], here we focus on the paradigm of Reservoir Computing, which allows to produce highly resource-efficient RNNs with a long-proven effectiveness in applications with sensor gathered information such as human activity recognition [7], ambient assisted living [8], medical diagnosis [9], meteorological forecasting [10], [11], industrial applications (for blast furnace off-gas) [12], [13], and more [14]. In all these cases, Reservoir Computing approaches provide an unparalleled tool in the literature both from the point of the achievable predictive performance, and of the trade-off between accuracy and efficiency in learning. We will show how the use of Echo State Networks (ESNs) [15], [16], from the Reservoir Computing paradigm [17], [18], allows the implementation of an optimal aggregation strategy that produces models equivalent to the corresponding centralized model directly trained on all the data. Nevertheless, with the proposed approach privacy preservation constraints are still satisfied.

This works has been developed within the context of the H2020 project TEACHING[1] [19] which is an ongoing research endeavour targeting specifically the provisioning of innovative methods and systems to enable the development of the next-generation of autonomous applications leveraging a learning system distributed [20] over a cyber-physical system (CPS). In this context, TEACHING puts forward a human-centric perspective on CPS intelligence based on a synergistic collaboration between the human and the cybernetic intelligence. In particular, it leverages human reactions as a driver for continual adaptation [21] and personalization of neural models deployed on the devices at the edge of the CPS. This clearly depicts a federated learning scenario where such localized neural models will be adapted across time to provide personalized predictions to the single users, while maintaining centralized and aggregated models on the cloud, leveraging the

[1] www.teaching-h2020.eu

This is a pre-print version of the following paper

Davide Bacciu, Daniele Di Sarli, Pouria Faraji, Claudio Gallicchio, Alessio Micheli: Federated Reservoir Computing Neural Networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN 2021), IEEE, Forthcoming.

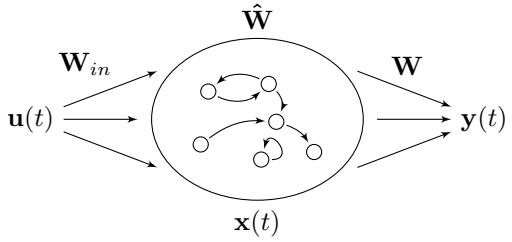

Fig. 1. Architecture of an ESN. The input signal $\mathbf{u}(t)$ is fed to the recurrent reservoir. Then, a state $\mathbf{x}(t)$ is extracted from the reservoir, from which an output $\mathbf{y}(t)$ is computed.

full knowledge harvested by the personalized models. Within such a scenario, TEACHING is planning to leverage ESNs as the model of choice to implement the learning models distributed at the CPS edge, using the federated learning mechanisms described in this paper to build, maintain and consolidate the aggregated models at the network core.

In Section II we introduce ESNs and the commonly used aggregation technique denoted as Federated Averaging. In Section III we propose a novel federation technique, denoted as Incremental Federated Learning, for producing aggregated ESN models that are equivalent to a centralized model. In Section IV we perform an experimental comparison of the approaches of Federated Averaging and Incremental Federated Learning, by simulating a federated scenario over two datasets for human state and activity recognition. Finally, in Section V we draw the conclusions of the study.

II. Background

A. Echo State Networks

Echo State Networks (ESNs) [15], [16] are an efficient ML approach for temporal data. They are part of the general framework of Recurrent Neural Networks (RNNs), but are based on the exploitation of the network activations from the point of view of a discrete-time dynamical system. The idea of studying the evolution of the recurrent network as a dynamical system is not unique to ESNs, but is shared under the hat of the so-called Reservoir Computing paradigm [17], [18]. While in this work we focus on ESNs, the techniques are also applicable to other Reservoir Computing models such as Liquid State Machines [22].

The architecture of an ESN consists in two components, which are illustrated in Fig. 1. One is the recurrent network that holds an internal state which evolves over the time steps, which is called the *reservoir*. The other component, the *readout*, is a linear layer that takes as input a state of the reservoir and emits a prediction. Formally, let $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ denote the state of a reservoir with N_R recurrent units at a given time step t. Then, the evolution of the state for an input sequence of vectors $\mathbf{u}(1), \ldots, \mathbf{u}(t) \in \mathbb{R}^{N_U}$ in a reservoir of leaky-integrator neurons can be described as

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{0}, \\ \mathbf{x}(t) &= (1-a)\,\mathbf{x}(t-1) \\ &\quad + a \tanh\left(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1)\right). \end{aligned} \tag{1}$$

Equation (1) is parametrized by two matrices and a scalar: matrix $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir-to-reservoir weight matrix, and $a \in \mathbb{R}$ is the leaking rate, under the constraint that $0 < a \leq 1$. The bias term is omitted for the sake of conciseness.

Unlike popular RNNs in which the parameters of the whole network are jointly trained by an interative algorithm, in ESNs only the parameters of the readout are trained. This allows for an extremely efficient training process. In fact, the weights in the reservoir are initialized from a suitable random distribution and then left fixed. For the reservoir-to-reservoir matrix $\hat{\mathbf{W}}$, the initialization step also includes an important constraint on the spectral radius $\rho(\hat{\mathbf{W}})$ (the largest eigenvalue in absolute value) which is controlled in order to meet the condition for the stability of the reservoir dynamics [15]. Moreover, $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are often initialized as sparse, i.e., to have a limited degree of connectivity between the units, to enable faster matrix operations.

From a given state $\mathbf{x}(t)$, the output of the network is computed by the readout as follows:

$$\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t). \tag{2}$$

Here, $\mathbf{W} \in \mathbb{R}^{N_Y \times N_R}$ is a weight matrix. In ESNs, $\mathbf{W}$ is the only matrix subject to training. As such, training proceeds as follows:

1) the input sequences from the training dataset are fed to the reservoir,
2) the relevant states on which the network must learn to perform predictions are collected column-wise into a matrix $\mathbf{S} \in \mathbb{R}^{N_R \times N_{train}}$, where N_{train} is the number of such states, and the associated targets are collected into the matrix $\mathbf{Y} \in \mathbb{R}^{N_Y \times N_{train}}$,
3) the matrix $\mathbf{W}$ is obtained as the solution to a least squares minimization problem between $\mathbf{WS}$ and $\mathbf{Y}$.

In particular, a common algorithm for a regularized solution to the least squares problem is ridge regression. In this case, if $\beta \in \mathbb{R}^+$ is the L2 regularization factor chosen by model selection, the readout weights are computed in closed form as follows:

$$\mathbf{W} = \mathbf{Y}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \beta\mathbf{I})^{-1}. \tag{3}$$

By avoiding the tuning of the recurrent connections, the training process of ESNs can be particularly efficient. Moreover, by avoiding the use of gradient descent it does not run into the optimization problems associated with the popular algorithm of backpropagation through time [23].

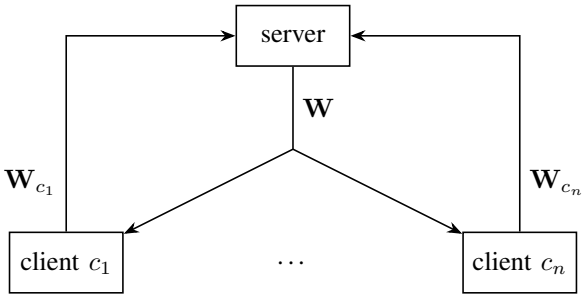

Fig. 2. Federated Averaging Scheme. Each client c sends their local matrix $\mathbf{W}_c$ to the server. After the aggregation of the models is performed in the form of a weighted average, the server sends back the same matrix $\mathbf{W}$ to all clients.

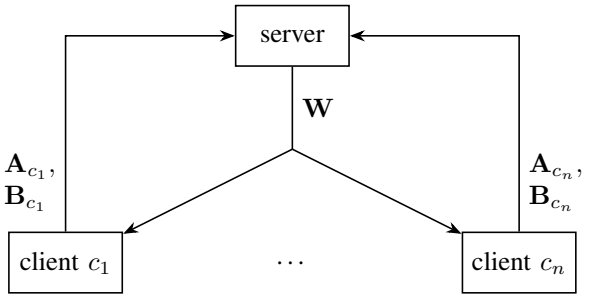

Fig. 3. Incremental Federated Learning Scheme. Each client sends their local matrices $\mathbf{A}_c$ and $\mathbf{B}_c$ to the server. After the matrices are aggregated and multiplied to compute the optimal readout weights, the server transmits $\mathbf{W}$ back to all clients.

B. Federated Averaging

A straightforward and standard strategy for performing federated learning in neural networks is that of Federated Averaging (FedAvg) [24]. In this strategy, the weights of each locally trained model are aggregated in the central server by an element-wise average, possibly weighted by the size of the local datasets.

In the special case of ESNs, we can assume the scenario of a uniform configuration of the reservoir among all clients. In practice this means that the input-to-reservoir matrices $\mathbf{W}_{in}$ and the reservoir-to-reservoir matrices $\hat{\mathbf{W}}$ will be identical in all clients. In this case, since $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are fixed, Federated Averaging simply amounts to the transmission and averaging of the readout weights alone.

Formally, let $\mathbf{S}_c \in \mathbb{R}^{N_R \times N_{train,c}}$ be a matrix containing the states collected from the reservoir, locally to client $c \in \mathcal{C}$. The states in $\mathbf{S}_c$ can be used to locally train the readout weights $\mathbf{W}_c$ in closed-form as follows:

$$\mathbf{W}_c = \mathbf{Y}_c \mathbf{S}_c^T (\mathbf{S}_c \mathbf{S}_c^T + \beta_c \mathbf{I})^{-1}, \tag{4}$$

where $\mathbf{Y}_c \in \mathbb{R}^{N_Y \times N_{train,c}}$ contains the label associated to each reservoir state in $\mathbf{S}_c$, $\beta_c \in \mathbb{R}^+$ is the L2 regularization factor and $\mathbf{I}$ is the identity matrix.

After the local readout weights $\mathbf{W}_c$ have been computed, they can be transferred to the server. In this case, the local readout weights are the only information which is sent to the server, which is not aware of the content of the actual data or states from the clients. The average weights are computed as follows:

$$\mathbf{W} = \sum_{c \in \mathcal{C}} \frac{1}{N_{train,c}} \mathbf{W}_c \tag{5}$$

After the aggregated weights are computed as in (5), the matrix $\mathbf{W}$ is transmitted back to the clients. Then, the clients can substitute their readout weights with the averaged version received from the server. A schematic view of how the matrices are transmitted between clients and server is shown in Fig. 2.

Let us ignore the impact of transmitting single scalar values such as $N_{train,c}$. Then, with the technique of Federated Averaging, each client needs to share with the server $N_Y N_R$ floating-point values, corresponding to the entries in $\mathbf{W}_c$. On the other hand, each client receives from the server $N_Y N_R$ floating-point values for $\mathbf{W}$. The transmission load is thus symmetric.

Averaging the readout weigths is a straightforward technique that however does not give any strong guarantee about the performance of the aggregated model. In the next section we will propose a different aggregation strategy that guarantees the optimal aggregated weights given the data and the reservoir.

III. Proposed method

The peculiar characteristics of ESN training allow an optimal form of federated learning, in the sense that the resulting aggregated model is equivalent to the model that would be obtained by aggregating all the input data and using it for the training process. In fact the proposed method, that we denote as *Incremental Federated Learning* (IncFed), exploits an algebraic decomposition of the typical readout training equation (3).

As for the Federated Averaging technique, this method also assumes a uniform configuration of the reservoir among all clients. Locally, instead of computing the readout weights, each client c computes the matrices $\mathbf{A}_c \in \mathbb{R}^{N_Y \times N_R}$ and $\mathbf{B}_c^{N_R \times N_R}$ as follows:

$$\mathbf{A}_c = \mathbf{Y}_c \mathbf{S}_c^T, \tag{6}$$

$$\mathbf{B}_c = \mathbf{S}_c \mathbf{S}_c^T + \beta_c \mathbf{I}. \tag{7}$$

The matrices $\mathbf{A}_c$ and $\mathbf{B}_c$ are sent to the server, where they get summed as in the following equations:

$$\mathbf{A} = \sum_{c \in \mathcal{C}} \mathbf{A}_c, \tag{8}$$

$$\mathbf{B} = \sum_{c \in \mathcal{C}} \mathbf{B}_c. \tag{9}$$

After the summed matrices are computed as in (8) and (9), the server can compute the optimal readout weights $\mathbf{W}$ in closed-form as follows:

$$\mathbf{W} = \mathbf{A}\mathbf{B}^{-1} \tag{10}$$

Notice how (10) is mathematically equivalent to (3) if all data was locally available to the server. After the weights are computed as in (10), they can be transmitted back to the clients as in the Federated Averaging approach from Section II-B. A schematic view of how the matrices are transmitted between clients and server in the *Incremental Federated Learning* approach is shown in Fig. 3.

The benefit of the proposed method with respect to the Federated Averaging is that it allows a one-shot training with virtually unlimited amount of data. The reason behind this is that both matrices $\mathbf{A}_c \in \mathbb{R}^{N_Y \times N_R}$ and $\mathbf{B}_c \in \mathbb{R}^{N_R \times N_R}$ do not depend on the number of training sequences. Whenever new data is available, the two matrices can be iteratively updated by the clients by simply adding the corresponding results from the associated computations. Formally, assume that the client c has already computed the matrices $\mathbf{A}_c^{(t)} \in \mathbb{R}^{N_Y \times N_R}$ and $\mathbf{B}_c^{(t)} \in \mathbb{R}^{N_R \times N_R}$ after t iterations. As soon as new input data is available together with the associated labels, the client can compute the matrices $\tilde{\mathbf{A}}_c$ and $\tilde{\mathbf{B}}_c$ like in 6 and 7 using only the newly available data, and then sum the matrices as in:

$$\mathbf{A}_c^{(t+1)} = \mathbf{A}_c^{(t)} + \tilde{\mathbf{A}}_c, \tag{11}$$

$$\mathbf{B}_c^{(t+1)} = \mathbf{B}_c^{(t)} + \tilde{\mathbf{B}}_c. \tag{12}$$

It can be easily noticed how equations (11) and (12) are equivalent to the direct computation of $\mathbf{A}_c$ and $\mathbf{B}_c$ with all the data. Then, the last iteration T can be selected for the matrices $\mathbf{A}_c \coloneqq \mathbf{A}_c^{(T)}$ and $\mathbf{B}_c \coloneqq \mathbf{B}_c^{(T)}$ to be sent to the server. It is this incremental nature of the approach that gives rise to the name of *Incremental Federated Learning*.

Even though the training equation is decomposed, and even though the server is aware of the reservoir weights that were used to produce the final states, we point out that the server is still unable to recover the original training data. In fact, the striking advantage of the proposed approach is that the training data is *never transferred* to other nodes and it is *not possible to recover it* from the transferred matrices, but at the same time the readout can be aggregated as if all the original training data was available to the server. Thus, *Incremental Federated Learning* enables an exact form of federation for ESNs while guaranteeing privacy preservation.

In *Incremental Federated Learning*, the number of floating-point values that get transferred from each individual client to the server is $N_R^2 + N_R N_Y$, while the transmission from the server to the clients involves $N_R N_Y$ floating-point values. The added transmission load with respect to *Federated Averaging* (which is still constant with respect to the number of examples used for training) is justified by a better predictive performance of the aggregated model, as it will be demonstrated in Section IV.

IV. Experimental evaluation

In ML applications governed by strict efficiency constraints, such as within low-power edge devices, the training process must be efficient. This is why the first choice in such contexts is often represented by ESNs. The aim of our experimental evaluation is to compare the performance of federated ESNs, on one side using the Federated Averaging approach, and on the other side using the proposed Incremental Federated Learning method. Our analysis will involve two datasets for human state and activity recognition, which are well-suited for simulating a federated scenario.

In the following we first discuss the datasets that have been used to evaluate the proposed Incremental Federated Learning approach. Then, we describe our experimental setup and we discuss the results of the experiments.

A. Datasets

To perform the experiments we have chosen two dataset whose data is organized on a per-subject basis. This makes the datasets suited for a comparison of federated learning techniques by assuming a different edge device for each subject in the datasets. In the following we briefly describe the two datasets.

a) WESAD: WESAD [25] is a publicly available dataset for stress and affect detection from wearable devices. The time-series in the dataset are recorded from both a wrist- and a chest-worn device, in a lab study comprising of 15 subjects. For our purposes we employ a subset of the data, in particular we only consider the following signals: *electrocardiogram*, *electrodermal activity*, *electromyogram*, *respiration*, *body temperature*, and *three-axis acceleration* for a total of 8 features. All considered signals are synchronized and sampled at 700 Hz. We consider the classification problem of predicting the state of the user from the aforementioned signals, restricted to the following 4 classes:

- Baseline
- Stress
- Amusement
- Meditation.

b) HAR: The Heterogeneity Activity Recognition dataset from smartphones and smartwatches sensors [26] is a dataset collected in real-world settings that can be used for benchmarking tasks of human activity recognition. The time-series contained in the dataset have been produced by sensors commonly found in smartphones, namely accelerometer, gyroscope, magnetometer and GPS. The data was collected by having subjects carry smartphones or smartwatches while performing scripted activities in no specific order. The dataset includes the following 6 kinds of activities:

- Biking
- Sitting
- Standing
- Walking
- Stair Up
- Stair Down.

For our purposes we only consider the signals coming from the *accelerometer* and the *gyroscope*, which are sampled at the highest frequency that the respective device allows, which is between 50 Hz and 200 Hz. Devices include 4 smartwatches and 8 smartphones, which record the data of 9 different subjects in total.

TABLE I
HYPER-PARAMETER RANGES EXPLORED FOR THE WESAD DATASET.

Units	Spectral radius	Leaking rate	Input scaling	Input connectivity	Recurrent connectivity
100, 250	[0.1, 0.99]	[0.1, 1]	[0.1, 1]	[1, 60]	[1, 60]

TABLE II
HYPER-PARAMETER RANGES EXPLORED FOR THE HAR DATASET.

Units	Spectral radius	Leaking rate	Input scaling	Input connectivity	Recurrent connectivity
100, 500	[0.1, 0.99]	[0.1, 1]	[0.1, 1]	[1, 60]	[1, 60]

B. Experimental setup

Our aim is to measure and compare the performance of the two different federation strategies, namely Federated Averaging and the proposed Incremental Federated Learning. To evaluate the strategies in different contexts, we simulate four different degrees of availability of the clients in the federation. In particular, we simulate the scenarios in which only 25%, 50%, 75% or 100% of the subjects are available from each dataset.

While the number of subjects in the training set is controlled in order to simulate different numbers of clients in the federation, we emphasize that the number of subject in the validation and test sets remains fixed in all experiments.

The best-performing hyperparameters, which are selected by evaluating on the validation set, are shared across all models in the federation. Therefore, the models within all clients share the same hyperparameters. Moreover, the models also share the same exact initialization for the weights in the reservoir (matrices $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$).

We point out that the practice of using all the training data to perform hyperparameter tuning is obviously not realistic in practical federated learning applications. As an alternative, one could explore how the models behave when the hyperparameters are chosen on the smallest fraction of the dataset. In this study we have chosen to use the entire training set for the hyperparameter search in order to limit the number of variables involved in our experimental evaluation.

The two datasets are processed as follows.

a) WESAD: The 8 synchronized time series that are considered from the WESAD dataset are split in chunks of 350 samples (roughly 0.5 seconds for each chunk) and each chunk is associated to its target class.

For WESAD, the hyperparameter tuning is performed by a hold-out evaluation strategy. In detail, the dataset is split so that 3 subjects ($\sim 20\%$) are used for the test set (used for testing the models after aggregation) and 3 other subjects ($\sim 20\%$) for the validation set (used for validating the models after aggregation). All other subjects are used for training.

In Table I we report the range of hyperparameters explored for each model on WESAD.

TABLE III
SELECTED HYPERPARAMETERS FOR THE FEDERATED AVERAGING AND INCREMENTAL FEDERATED LEARNING APPROACHES ON THE WESAD DATASET.

	FedAvg	IncFed
Units	100	100
Spectral radius	0.9	0.99
Leaking rate	1	1
Input scaling	0.1	1
Input connectivity	50	5
Recurrent connectivity	1	50
Training Accuracy (%)	75.5	85.08
Validation Accuracy (%)	84.03	83.63

TABLE IV
SELECTED HYPERPARAMETERS FOR THE FEDERATED AVERAGING AND INCREMENTAL FEDERATED LEARNING APPROACHES ON THE HAR DATASET.

	FedAvg	IncFed
Units	500	500
Spectral radius	0.8	0.4
Leaking rate	1	1
Input scaling	0.5	1
Input connectivity	20	50
Recurrent connectivity	20	60
Training Accuracy (%)	82.17	94.53
Validation Accuracy (%)	75.74	79.99

b) HAR: From the HAR dataset we only select the time-series associated to the accelerometer and to the gyroscope. For each of these, the dataset specified the values for each of the three axes (*x*, *y*, *z*), for a total of 6 features. Additional features such as the mean, standard deviation, minimum and maximum of the aforementioned 6 features are computed. This is done by using a sliding window of size 200 with 50% overlap, as described in detail in [27]. Moreover, we also include as additional features the magnitudes for the accelerometer and gyroscope data. Therefore, the resulting data used to train the models is composed by a total of 32 features.

Regarding the length of the sequences, we have chosen to split the time-series into sequences of length 500.

The hyperparameter tuning is performed by a hold-out evaluation strategy. In detail, the dataset is split so that 2 subjects ($\sim 20\%$) are used for the test set (used for testing the models after aggregation) and 2 other subjects ($\sim 20\%$) for the validation set (used for validating the models after aggregation). All other subjects are used for training.

In Table II we report the range of hyperparameters explored for each model on HAR.

C. Results

In Table III we report the best performing hyperparameters, chosen on the validation set, for Federated Averaging and Incremental Federated Learning on the WESAD task. The corresponding results for the HAR task are reported in Table IV. As it can be observed from Table III and Table IV, the accuracy on the validation set is similar for both approaches.

TABLE V
TRAINING AND TEST ACCURACY FOR WESAD USING FEDERATED AVERAGING AND INCREMENTAL FEDERATED LEARNING.

Training subjects	Random baseline		Centralized ESN		FedAvg ESN		IncFed ESN	
	Training	Test	Training	Test	Training	Test	Training	Test
25%	25.30	24.59	93.14 ± .04	**65.96** ± .08	81.55 ± .09	63.42 ± .10	93.14 ± .04	**65.96** ± .08
50%	25.27	25.45	87.51 ± .06	**74.89** ± .09	77.63 ± .09	71.75 ± .11	87.51 ± .06	**74.89** ± .09
75%	25.09	24.32	84.45 ± .04	**76.56** ± .07	76.77 ± .11	74.69 ± .09	84.45 ± .04	**76.56** ± .07
100%	24.91	24.99	83.78 ± .06	**77.92** ± .06	76.57 ± .12	75.81 ± .10	83.78 ± .06	**77.92** ± .06

TABLE VI
TRAINING AND TEST ACCURACY FOR HAR USING FEDERATED AVERAGING AND INCREMENTAL FEDERATED LEARNING.

Training subjects	Random baseline		Centralized ESN		FedAvg ESN		IncFed ESN	
	Training	Test	Training	Test	Training	Test	Training	Test
25%	16.65	15.77	92.73 ± .00	55.96 ± .01	99.17 ± .00	**60.38** ± .02	92.73 ± .00	55.96 ± .01
50%	16.82	17.13	93.60 ± .01	**70.28** ± .03	81.57 ± .07	66.99 ± .09	93.60 ± .01	**70.28** ± .03
75%	16.53	17.54	93.84 ± .02	**75.69** ± .05	78.52 ± .07	74.86 ± .06	93.84 ± .02	**75.69** ± .05
100%	16.75	16.99	93.64 ± .04	**80.19** ± .03	70.18 ± .07	74.84 ± .11	93.64 ± .04	**80.19** ± .03

To evaluate the actual generalization performance of the two approaches we have tested the models on the held-out test set. The results for WESAD and HAR are reported respectively in Table V and in Table VI, where we show the average accuracy and standard deviation over 3 repetitions of the experiments. The tables show the results of the experiments with a varying number of subjects, or clients, in the federation. In particular, the reported values on the training and test set are measured *after* aggregation (where applicable). In federation terms, the test accuracy indicates the average performance of the aggregated model for a set of clients that join the federation at a later time.

In the tables, the accuracies of a random baseline are included for reference. Since the distribution of the classes is balanced, the random model sets a baseline over which to evaluate how much the other models are actually learning from the data. Note that for clarity, and for numerical confirmation, we also report the performance obtained by a centralized ESN, which turns out to be perfectly compatible with those achieved by the Incremental Federated Learning ESN, as it is clearly reflected in the values. As highlighted in bold, the reader can observe from Table I and Table II that the Incremental Federated Learning method has superior predictive performance with respect to the Federated Averaging in all cases except just one.

Still from Tables I and II, it is interesting to observe how both training and test accuracies vary with the number of subjects in the training set. For Federated Averaging, increasing the number of clients can drastically decrease the training accuracy. This can be interpreted as a form of overfitting in the case of few subjects: in fact, for a high training accuracy we can observe an associated poor generalization performance on the test set. On the other hand, an increasing accuracy on the test set for increasing numbers of subjects is to be expected and it is clearly highlighted by our results.

V. CONCLUSIONS

Thanks to their efficient training process ESNs and, in general, Reservoir Computing approaches, are often considered ideal for deployment on low-powered devices such as those that can be found in the edge. In this work we have shown that it is possible to further exploit the peculiar training method of ESNs to improve their predictive accuracy in a federated learning scenario. In particular, the novel *Incremental Federated Learning* approach that we propose makes Reservoir Computing models such as ESNs especially suited for federated infrastructures.

The advantages of the proposed approach are manifold. First, the long-proven characteristics of ESNs make it possible to train predictive models *very efficiently*, even directly on the edge. Second, the global model that is produced by aggregating the local models is *optimal* in the sense that no better equivalent model could have been produced by gathering all the training data within a centralized node. Third, *privacy constraints* are preserved since the potentially sensitive training data is never transmitted over the network and remains confined within each local node.

We point out that the proposed approach is not limited to a specific architecture of the reservoir. While here for simplicity we have employed an ESN with leaky-integrator neurons, the approach can be extended without modifications to more complex reservoirs (e.g. Deep ESNs [28]) that for their characteristics are often better suited for modeling multiple time-scales in the data. This makes Incremental Federated Learning a highly versatile federation method for Reservoir Computing models.

ACKNOWLEDGMENT

This work is supported by the EC H2020 programme under project TEACHING (grant n. 871385).

REFERENCES

[1] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[2] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018.

[3] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private language models without losing accuracy," *CoRR*, vol. abs/1710.06963, 2017.

[4] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *CoRR*, vol. abs/1906.04329, 2019.

[5] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *ICLR*. OpenReview.net, 2020.

[6] R. A. Sater and A. B. Hamza, "A federated learning approach to anomaly detection in smart buildings," *CoRR*, vol. abs/2010.10293, 2020.

[7] F. Palumbo, C. Gallicchio, R. Pucci, and A. Micheli, "Human activity recognition using multisensor data fusion based on reservoir computing," *J. Ambient Intell. Smart Environ.*, vol. 8, no. 2, pp. 87–107, 2016.

[8] D. Bacciu, C. Gallicchio, A. Micheli, S. Chessa, and P. Barsocchi, "Predicting user movements in heterogeneous indoor environments by reservoir computing," in *Proc. of the IJCAI Workshop on Space, Time and Ambient Intelligence (STAMI), Barcellona, Spain*. Citeseer, 2011, pp. 1–6.

[9] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep echo state networks for diagnosis of parkinson's disease," in *ESANN*, 2018.

[10] M. Alizamir, S. Kim, O. Kisi, and M. Zounemat-Kermani, "Deep echo state network: a novel machine learning approach to model dew point temperature using meteorological variables," *Hydrological Sciences Journal*, vol. 65, no. 7, pp. 1173–1190, 2020.

[11] T. Kim and B. R. King, "Time series prediction using deep echo state networks," *Neural Computing and Applications*, vol. 32, no. 23, pp. 17 769–17 787, 2020.

[12] S. Dettori, I. Matino, V. Colla, and R. Speets, "Deep echo state networks in industrial applications," in *AIAI (2)*, ser. IFIP Advances in Information and Communication Technology, vol. 584. Springer, 2020, pp. 53–63.

[13] V. Colla, I. Matino, S. Dettori, S. Cateni, and R. Matino, "Reservoir computing approaches applied to energy management in industry," in *EANN*, ser. Communications in Computer and Information Science, vol. 1000. Springer, 2019, pp. 66–79.

[14] B. Schrauwen, D. Verstraeten, and J. M. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *ESANN*, 2007, pp. 471–482.

[15] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks – with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 2001.

[16] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[17] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.

[18] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.

[19] D. Bacciu, S. Akarmazyan, E. Armengaud, M. Bacco, G. Bravos, C. Calandra, E. Carlini, P. Cassarà, M. Coppola, P. Dazzi, M. C. Degennaro, D. Di Sarli, J. Dobaj, C. Gallicchio, S. Girbal, A. Gotta, R. Groppo, G. Macher, D. Mazzei, G. Mencagli, D. Michail, A. Micheli, F. Odierna, R. Peroglio, S. Petroni, R. Potenza, F. Pourdanesh, K. Tserpes, F. Tagliabò, J. Valtl, I. Varlamis, and O. Veledar, "Teaching - trustworthy autonomous cyber-physical applications through human-centred intelligence," in *Submitted*, 2021.

[20] D. Bacciu, S. Chessa, C. Gallicchio, A. Lenzi, A. Micheli, and S. Pelagatti, "A general purpose distributed learning model for robotic ecologies," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 435 – 440, 2012, 10th IFAC Symposium on Robot Control.

[21] A. Cossu, A. Carta, and D. Bacciu, "Continual learning with gated incremental memories for sequential data processing," in *Proceedings of the 2020 IEEE World Congress on Computational Intelligence*, 2020.

[22] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.

[23] Y. Bengio, P. Y. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282.

[25] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, "Introducing WESAD, a multimodal dataset for wearable stress and affect detection," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ser. ICMI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 400–408. [Online]. Available: https://doi.org/10.1145/3242969.3242985

[26] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, and M. R. Luaces, "A public domain dataset for real-life human activity recognition using smartphone sensors," *Sensors*, vol. 20, no. 8, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/8/2200

[27] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2018, pp. 1103–1111. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/BDCloud.2018.00164

[28] C. Gallicchio, A. Micheli, and L. Pedrelli, "Design of deep echo state networks," *Neural Networks*, vol. 108, pp. 33–47, 2018.