Decentralized Plasticity in Reservoir Dynamical Networks for Pervasive Environments

Valerio De Caro [1] **Davide Bacciu** [1] **Claudio Gallicchio** [1]

Abstract

We propose a framework for localized learning with Reservoir Computing dynamical neural systems in pervasive environments, where data is distributed and dynamic. We use biologically plausible intrinsic plasticity (IP) learning to optimize the non-linearity of system dynamics based on local objectives and extend it to account for data uncertainty. We develop two algorithms for federated and continual learning, FedIP and FedCLIP, which respectively extend IP to client-server topologies and prevent catastrophic forgetting in streaming data scenarios. Results on real-world datasets from human monitoring show that our approach improves performance and robustness, while preserving privacy and efficiency.

1. Introduction

The increasing demand for ML systems in on-the-edge applications (Bacciu et al., 2021a; De Caro et al., 2022) poses new challenges for learning in pervasive environments, where large numbers of resource-constrained devices are involved (Figure 1). For example, in healthcare applications (Nguyen et al., 2022; Can & Ersoy, 2021), physiological data from wearable devices must be processed to detect heart conditions, while respecting privacy regulations (Horvitz & Mulligan, 2015) and ensuring model reliability over time.

We identify three main challenges for learning in this domain: (1) achieving a good trade-off between performance and efficiency on temporal data; (2) complying with privacy constraints that prevent data sharing; (3) avoiding *data oblivion*, i.e., the loss of information due to data discarding.

Existing learning methodologies can partially address these challenges. Echo State Networks (ESNs) (Jaeger, 2001) are efficient models for temporal data that have been successful

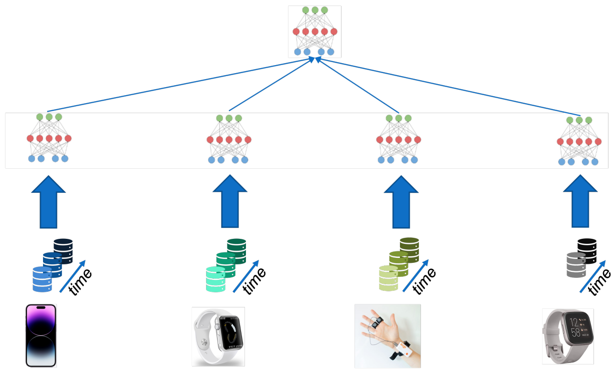

Figure 1. Learning in a pervasive environment.

in Human Activity Recognition (HAR) applications (Bacciu et al., 2021b). Federated Learning (FL) (McMahan et al., 2017) is a distributed learning method that preserves data privacy by learning a global model without transferring local data. Continual Learning (CL) (Parisi et al., 2019) is a learning paradigm that allows updating a model over time from a continuous data stream without forgetting previous knowledge. However, the integration of these areas is still limited and none of the existing works address the scenario we consider in this paper.

We propose a methodology and practical algorithms for learning in pervasive environments based on Intrinsic Plasticity (Triesch, 2005), an unsupervised algorithm for adapting a reservoir's dynamics to the input sequence. Our contribution is threefold: (1) we extend the learning approach of Intrinsic Plasticity (`IP`) to handle the uncertainty arising from data distribution over space and time; (2) we propose Federated Intrinsic Plasticity (`FedIP`), an instantiation of the Federated Averaging algorithm for adapting a reservoir from client-server federation with stationary data; (3) we introduce Federated Continual Intrinsic Plasticity (`FedCLIP`), an extension of `FedIP` to deal with non-stationary scenarios. We evaluate the algorithms with an incremental experimental setup based on two HAR benchmarks and show that they can improve the performance of the global model with low computation and communication overhead, and cope with data non-stationarity.

[1]Department of Computer Science, University of Pisa, Pisa, Italy. Correspondence to: Valerio De Caro <valerio.decaro@phd.unipi.it>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

2. Local Dynamics Adaptation in Pervasive Environments

Reservoir Computing Reservoir Computing (RC) (Lukoševičius & Jaeger, 2009) is a paradigm that leverages the evolution of the neural activations of Recurrent Neural Networks (RNNs) as a discrete-time non-linear dynamical system. A remarkable example in RC is represented by ESNs (Jaeger, 2001; Jaeger & Haas, 2004), which allow learning on sequential data efficiently. ESNs are made up of two main components: a *reservoir*, a recurrent layer of sparsely connected neurons, holding the internal state which evolves over time; a *readout*, a typically linear transformation on the domain of the reservoir states. Formally, we consider an ESN with N_U input units, N_R hidden recurrent units, and N_Y output units. Given an input sequence of vectors $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ with $t \in \{0, \ldots, T-1\}$, the equations modeling the state transition of the reservoir with leaky-integrator neurons (Jaeger et al., 2007) and the transformation applied by the readout can be described as

$$\begin{aligned} \mathbf{x}_{net}(t) &= \mathbf{W}_{in}\mathbf{u}(t) + \mathbf{b}_{rec} + \hat{\mathbf{W}}\mathbf{x}(t-1), \\ \mathbf{x}(t) &= (1-a)\,\mathbf{x}(t-1) + af\left(\mathbf{x}_{net}(t)\right), \\ \mathbf{y}(t) &= \mathbf{W}\mathbf{x}(t) + \mathbf{b}_{out}, \end{aligned} \tag{1}$$

where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ 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, $\mathbf{b}_{rec} \in \mathbb{R}^{N_R}$ is the reservoir bias term, f is the non-linearity applied to the neurons' cumulative input, $a \in (0, 1]$ is the leaking rate, $\mathbf{W} \in \mathbb{R}^{N_Y \times N_R}$ is the readout weight matrix, $\mathbf{b}_{out}$ is the output bias term. The hidden state of the reservoir at time $t = 0$ is initialized as $\mathbf{x}(0) = \mathbf{0}$.

Instead of backpropagating the error signal through time as in standard RNNs, ESNs keep the input-to-reservoir matrix $\mathbf{W}_{in}$ and the reservoir-to-reservoir matrix $\hat{\mathbf{W}}$ **fixed**, with the only constraint of choosing spectral radius $\rho(\hat{\mathbf{W}}) < 1$ to empirically display asymptotically stable dynamics[1]. Thus, the readout weights $\mathbf{W}$ are the only free parameters. By formulating the learning problem of the readout as a least squares problem, we can take advantage of Ridge Regression (RR) to obtain a closed-form solution defined as

$$\mathbf{W} = \mathbf{Y}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \lambda\mathbf{I})^{-1}, \tag{2}$$

where $\mathbf{Y}$ denotes the matrix of time-ordered target labels, $\mathbf{S}$ is the matrix of time-ordered reservoir states, λ is an L2-regularization term, and $\mathbf{I}$ is the identity matrix.

Intrinsic Plasticity Given the formulation of the reservoir, it is clear that the random initialization may affect the representation capabilities of the dynamical system. Following the line of argument from (Schrauwen et al., 2008), maximizing the information gain of a reservoir corresponds to maximizing the entropy of the output distribution of each neural unit. When the non-linearity f is the $\tanh$ function, the maximum entropy distribution is the Gaussian. As a result, the information gain can be measured as the following Kullback-Leibler divergence:

$$\begin{aligned} \mathcal{L}(\theta; \mu, \sigma) &= D_{KL}(\tilde{q}\,||\,\mathcal{N}_{\mu,\sigma}) \\ &= \int \tilde{q}(x) \log\left(\frac{\tilde{q}(x)}{\mathcal{N}_{\mu,\sigma}(x)}\right) \mathrm{d}x, \end{aligned} \tag{3}$$

where $\tilde{q}$ is the empirical distribution of the neural activations upon application of $\tilde{f}(\cdot\,; \theta)$ as non-linearity, and $\mathcal{N}_{\mu,\sigma}$ is the desired Gaussian distribution with mean μ and standard deviation σ. Given the formula in eq. (3), minimizing the Kullback-Leibler (KL) divergence between the empirical distribution and the desired Gaussian corresponds to maximizing the information gain of a reservoir. From a practical perspective, this is performed by IP (Schrauwen et al., 2008), an algorithm inspired by a biological phenomenon called *homeostatic plasticity*. Focusing the attention on the i-th neural unit, the algorithm requires the neuron's function to be reformulated as $\tilde{f}(x^i_{net}; \theta^i) = f(g^i x^i_{net} + b^i)$, where $\theta = \{\mathbf{g}, \mathbf{b}\}$ is the set of learnable, unit-wise gain and the bias parameters of the reservoir's non-linearity. The derivation of the loss function leads to the following update rules for the set of gain and bias parameters:

$$\begin{aligned} \Delta b &= -\eta\left(-\frac{\mu}{\sigma^2} + \frac{\tilde{x}}{\sigma^2} + 1 - \tilde{x}^2 + \mu\tilde{x}\right), \\ \Delta g &= \frac{\eta}{g} + \Delta b x_{net}, \end{aligned} \tag{4}$$

where $\tilde{x}$ is the result of the application of $\tilde{f}$ in the computation of the state transition, and η is the learning rate. A desirable effect of the algorithm is the reduction of the variance in performance caused by bad initializations of the reservoir's parameters.

The objective of our work is to extend this formulation to cope with decentralized environments characterized by both stationary and non-stationary data distributions.

2.1. Decentralized Plasticity with Stationary Data

Dealing with an environment that is characterized by a massive distribution of devices with private data leads straightforwardly to formulating the problem as a Federated Learning (FL) problem (McMahan et al., 2017). FL is a distributed machine learning approach that aims to solve learning tasks by a loose federation of participating devices while complying with privacy requirements of their local data (Mothukuri et al., 2021). In particular, it learns a global model by aggregating only local models, instead of gathering raw data from participants. Formally, given a finite

[1]The interested reader can find rigorous discussions on theoretical aspects of reservoir initialization in (Yildiz et al., 2012; Gallicchio & Micheli, 2017).

set clients $\mathcal{C}$, and assuming that each client $c \in \mathcal{C}$ holds a local dataset $\mathcal{D}_c$, a FL problem consists in learning a global model f by minimizing the following objective function:

$$\mathcal{L}(f) = \sum_{c \in \mathcal{C}} p_c \mathcal{L}_c(f_c, \mathcal{D}_c), \tag{5}$$

where f_c denotes the local realization of the global model, $\mathcal{L}_c(f_c, \mathcal{D}_c)$ is the local loss function computed on the local dataset, and p_c are client-specific weighting factors such that $0 \leq p_c \leq 1$ and $\sum_{c=1}^{|\mathcal{C}|} p_c = 1$. The parameters p_c have a twofold purpose: (1) selecting the clients to involve in the learning process; (2) weighing local loss functions to account for the non-i.i.d. data distribution among the clients (*statistical heterogeneity*) and the different availability and reliability of the devices where they are deployed (*system heterogeneity*).

In the federated setting it is reasonable to assume that, given a finite set of clients $\mathcal{C}$, we compute the global model θ as a convex combination of the local models learned by the clients, i.e., $\theta = \sum_{c \in \mathcal{C}} p_c \theta_c$. Here, we specialize this concept by assuming that the distribution $\tilde{q}$ can be calculated as a convex combination of the clients' local distributions, i.e., $\tilde{q} = \sum_{c \in \mathcal{C}} p_c \tilde{q}_c$. This assumption allows us to derive an upper bound of the centralized loss function in eq. (3):

$$\begin{aligned} D_{KL}(\tilde{q} \,||\, \mathcal{N}_{\mu,\sigma}) &= D_{KL}\left(\sum_{c \in \mathcal{C}} p_c \tilde{q}_c \,||\, \mathcal{N}_{\mu,\sigma}\right) \\ &= D_{KL}\left(\sum_{c \in \mathcal{C}} p_c \tilde{q}_c \,||\, \sum_{c \in \mathcal{C}} p_c \mathcal{N}_{\mu,\sigma}\right) \\ &\leq \sum_{c \in \mathcal{C}} p_c D_{KL}(\tilde{q}_i \,||\, \mathcal{N}_{\mu,\sigma}). \end{aligned}$$

Since minimizing the upper bound implies the minimization of the initial loss, we can re-define the loss function for the federated setting as follows:

$$\begin{aligned} \mathcal{L}_F(\theta; \mu, \sigma) &= \sum_{c \in \mathcal{C}} p_c \mathcal{L}_c(\theta_c; \mu, \sigma) \\ &= \sum_{c \in \mathcal{C}} p_c D_{KL}(\tilde{q}_c \,||\, \mathcal{N}_{\mu,\sigma}), \end{aligned} \tag{6}$$

where $\tilde{q}$ is the global distribution of the reservoir's neural activations, $\tilde{q}_c$ is the empirical distribution of the activations of client c, computed with its local realization of the global model $\tilde{f}_c$ on the local data $\mathcal{D}_c$, and p_c is the weighting factor introduced in eq. (5). In this formulation of the loss function, the parameters p_c account for the same purpose of eq. (5), balancing the contributions from clients depending on the statistical and system heterogeneity.

Federated Intrinsic Plasticity Our proposal is intended for a client-server topology and is based on Federated Averaging (`FedAvg`). The algorithm, namely `FedIP` (initially proposed by (De Caro et al., 2022)), instantiates `FedAvg` ((McMahan et al., 2017), described in 5) to learn the global gain and bias parameters, i.e., $\theta = \{\mathbf{g}, \mathbf{b}\}$, by minimizing the loss function in eq. (6) (the pseudocode of the server side is summarized in Algorithm 1). In the initialization phase, the server initializes reservoirs parameters $\mathcal{R} = \{\mathbf{W}_{in}, \hat{\mathbf{W}}, \mathbf{b}_{rec}, \alpha\}$ and broadcasts them to all the clients. Then, it initializes the global gain and bias parameters θ and begins the learning rounds. In a generic round r, the server broadcasts the global gain and bias θ^r to all the clients (line 6). This initializes the clients' side of the algorithm (summarized in Algorithm 2: each client $c \in \mathcal{C}$ performs E iterations of `IP` on the local dataset. Then, they send their local parameters $\theta_c^{r+1} = \{\mathbf{g}_c^{r+1}, \mathbf{b}_c^{r+1}\}$ back to the server (line 7). Finally, the server updates the parameters to θ^{r+1} by applying the same aggregation rule described in Algorithm 5 (line 9). In `FedIP`, the number of parameters exchanged between a client and the server in each communication step is exactly $2N_R$, with N_R number of neural units in the reservoir. A linear communication cost on the number of parameters of the model is a clear advantage in the federated setting and makes the algorithm suitable for scenarios in which we leverage far-edge devices. `FedIP` suits the general, gradient-based framework of `FedOpt` (Reddi et al., 2021) and allows for applying different client sam-

Algorithm 1 Federated Intrinsic Plasticity (`FedIP`)

```
Input: clients $\mathcal{C}$, number of rounds $R$, learning rate $\eta$, local epochs $E$, batch size $B$
$\mathcal{R} \leftarrow \{\mathbf{W}_{in}, \hat{\mathbf{W}}, \mathbf{b}_{rec}, \alpha\}$
$\theta^0 \leftarrow \{\mathbf{g}^0 \leftarrow \mathbf{1}, \mathbf{b}^0 \leftarrow \mathbf{0}\}$
Send $\mathcal{R}$ to all clients $c \in \mathcal{C}$
for each round $r \in \{0, 1, \ldots, R-1\}$ do
    for each client $c \in \mathcal{C}$ in parallel do
        Send $\theta^r$ to client $c$
        $\theta_c^r \leftarrow$ IPUpdate$_c$($\mathbf{g}^r$, $\mathbf{b}^r$, $\eta$, $E$)
    end for
    $\theta^{r+1} \leftarrow \left\{\sum_{c \in \mathcal{C}} \frac{n_c}{n} \mathbf{g}_c^{r+1}, \sum_{c \in \mathcal{C}} \frac{n_c}{n} \mathbf{b}_c^{r+1}\right\}$
end for
```

Algorithm 2 IPUpdate (on client c)

```
Input: global gain $\mathbf{g}^r$, global bias $\mathbf{b}^r$, learning rate $\eta$, local epochs $E$
$\mathbf{g}_c, \mathbf{b}_c \leftarrow \mathbf{g}^r, \mathbf{b}^r$
Split local data into a set of batches $\mathcal{B}$ of size $B$
for epoch $e \in \{0, 1, \ldots, E-1\}$ do
    for batch $b \in \mathcal{B}$ do
        Compute the average $\Delta\mathbf{g}^b$, $\Delta\mathbf{b}^b$ over $b$
        $\mathbf{g}_c, \mathbf{b}_c \leftarrow \mathbf{g}_c + \Delta\mathbf{g}^b, \mathbf{b}_c + \Delta\mathbf{b}^b$
    end for
end for
return $\mathbf{g}_c$, $\mathbf{b}_c$
```

pling (Huang et al., 2020; Lattimore & Szepesvári, 2020; Huang et al., 2021), local learning (Li et al., 2020; Karimireddy et al., 2020), and aggregation (Wang et al., 2020b;a; Lin et al., 2020) techniques.

2.2. Decentralized Plasticity with Non-Stationary Data

Pervasive environments are characterized by resource-constrained devices, and the data may not be retained for long periods of time due to such constraints. As a result, the learning process of each client unfolds on a data stream, and we cannot assume stationarity of the data over the stream.

In a non-stationary setting, we rely on the same formalization by (Lesort et al., 2021): we assume that, given a finite set of contexts $\mathcal{K}$ (i.e., possible states of the data distribution), there exists a hidden, discrete stochastic process $\{K_t\}_{t=1}^{T}$ that determines the evolution of the data distribution over time. Given that $p_{k,t}$ corresponds to the realization of the context variable for context k at time t, we can apply the following derivation:

$$\begin{aligned}
D_{KL}(\tilde{q}\,||\,\mathcal{N}_{\mu,\sigma}) &= \int \tilde{q}(x)\log\frac{\tilde{q}(x)}{\mathcal{N}_{\mu,\sigma}}\,\mathrm{d}x \\
&= \int \sum_{t=1}^{T}\sum_{k\in\mathcal{K}} p_{t,k}\tilde{q}_k(x)\log\frac{\sum_{k\in\mathcal{K}} p_{t,k}\tilde{q}_k(x)}{\mathcal{N}_{\mu,\sigma}}\,\mathrm{d}x \\
&= \sum_{t=1}^{T}\int \sum_{k\in\mathcal{K}} p_{t,k}\tilde{q}_k(x)\log\frac{\sum_{k\in\mathcal{K}} p_{t,k}\tilde{q}_k(x)}{\sum_{k\in\mathcal{K}} p_{t,k}\mathcal{N}_{\mu,\sigma}}\,\mathrm{d}x \\
&= \sum_{t=1}^{T} D_{KL}\left(\sum_{k\in\mathcal{K}} p_{t,k}\tilde{q}_k\,||\,\sum_{k\in\mathcal{K}} p_{t,k}\mathcal{N}_{\mu,\sigma}\right) \\
&\leq \sum_{t=1}^{T}\sum_{k\in\mathcal{K}} p_{t,k} D_{KL}(\tilde{q}_k\,||\,\mathcal{N}_{\mu,\sigma}).
\end{aligned}$$

However, the realizations of the context variable $p_{k,t}$ are not observable, and we can rely only on the information provided by the data available at time t. Thus, by assuming that, at time t, the available data are enough to approximate the behavior of the stochastic process, we can approximate the loss function as follows:

$$\begin{aligned}
\mathcal{L}_C(\theta;\sigma,\mu) &= \sum_{t=1}^{T}\sum_{k\in\mathcal{K}} p_{t,k} D_{KL}(\tilde{q}_k\,||\,\mathcal{N}_{\mu,\sigma}) \\
&\simeq \sum_{t=1}^{T} D_{KL}(\tilde{q}_t\,||\,\mathcal{N}_{\mu,\sigma}),
\end{aligned} \tag{7}$$

where $\tilde{q}_t$ is the empirical distribution computed upon application of $\tilde{f}(\cdot,\theta)$ on the dataset at experience t, i.e., $\mathcal{D}_t$, and $\mathcal{N}_{\mu,\sigma}$ is the desired gaussian distribution with mean μ and standard deviation σ.

Given a federated and non-stationary setting with a finite set of clients $\mathcal{C}$, each with its own bounded data stream of T learning experiences, we can plug the formulation with non-stationary data in the federated setting defined in 5, and define the loss function of `IP` as

$$\mathcal{L}_{FC}(\theta;\sigma,\mu) = \sum_{t=1}^{T}\sum_{c\in\mathcal{C}} p_{t,c} D_{KL}(\tilde{q}_{t,c}\,||\,\mathcal{N}_{\mu,\sigma}),$$

where $\tilde{q}_{t,c}$ is the empirical distribution computed by the client c on the local data observed at time t, i.e., $\mathcal{D}_{t,c}$ and $p_{t,c}$ is a weighting factor which balances the contributions from the clients at time t. Here, the weighting factor $p_{t,c}$ models the evolution of the data stream and system for each client. This implies that both statistical heterogeneity and system heterogeneity may evolve over time. In the former case, this depends on the relationship present between the stochastic processes of the clients involved. In the second case, it depends on the environmental conditions under which the process unfolds (e.g., device failures, mobility degrading network quality).

Federated Continual Intrinsic Plasticity The main issue that arises when learning from a stream of non-stationary data is phenomenon of **catastrophic forgetting** (McCloskey & Cohen, 1989; French, 1999): when exposing the model to learn multiple tasks sequentially, optimizing the model to solve the new task *interferes* with the knowledge about the previous tasks. Such a problem is addressed by CL (Parisi et al., 2019), which focuses on developing learning models capable of integrating novel information while mitigating interference with consolidated knowledge.

In a CL scenario, we assume that the data arrives in a streaming fashion as a sequence of learning *experiences* $S = e_1, e_2, \ldots$. Given a stream of n experiences $S = e_1, \ldots, e_n$, each experience e_i consists in a batch of examples $\mathcal{D}_i$. The objective of a CL algorithm is to minimize the loss $\mathcal{L}_S$ over the entire stream of data S:

$$\mathcal{L}_S(f_n^{CL}, n) = \frac{1}{\sum_{i=1}^{n}|\mathcal{D}_{test}^{i}|}\sum_{i=1}^{n}\mathcal{L}_{exp}(f_n^{CL}, \mathcal{D}_i^{test}), \tag{8}$$

where $\mathcal{L}_{exp}(f_n^{CL}, \mathcal{D}_i^{test})$ the loss computed on the test data of experience e_i, i.e., $\mathcal{D}_i^{test}$. This is usually performed with the aid of a memory buffer $\mathcal{M}$ which retains the knowledge from previous experiences in some form, and a continual learning strategy, which determines how to update $\mathcal{M}$.

To cope with this setting, we propose Federated Continual Intrinsic Plasticity (`FedCLIP`), which extends the local adaptation of clients for including a CL strategy (the pseudocode is summarized in Algorithms 3). `FedCLIP` relies on two assumptions: (1) all the clients face the same number of learning experiences, and the server is informed about such number; (2) at a generic time t, the clients are all synchronized on the learning experience $e_{t,c}$. The algorithm is

Algorithm 3 Federated Continual Intrinsic Plasticity (`FedCLIP`)

```
Input: clients $\mathcal{C}$, learning rate $\eta$, local epochs $E$, batch size $B$
$\mathcal{R} \leftarrow \{\mathbf{W}_{in}, \hat{\mathbf{W}}, \alpha\}$
$\theta_0^0 \leftarrow \{\mathbf{g}_0^0 \leftarrow \mathbf{1},\ \mathbf{b}_0^0 \leftarrow \mathbf{0}\}$
Send $\mathcal{R}$ to all clients $c \in \mathcal{C}$
for each experience $t \in \{0, 1, \dots, T-1\}$ do
    for each round $r \in \{0, 1, \dots, R-1\}$ do
        for each client $c \in \mathcal{C}$ in parallel do
            Send $\theta_t^r$ to client $c$
            $\theta_{t,c}^r \leftarrow$ ContinualIPUpdate$_c$($t$, $\mathbf{g}_t^r$, $\mathbf{b}_t^r$, $\eta$, $E$, $r == R-1$)
        end for
        $\theta_t^{r+1} \leftarrow \left\{ \sum_{c \in \mathcal{C}} \frac{n_{t,c}}{n_t} \mathbf{g}_{t,c}^r,\ \sum_{c \in [C]} \frac{n_{t,c}}{n_t} \mathbf{b}_{t,c}^r \right\}$
    end for
end for
```

Algorithm 4 ContinualIPUpdate (on client c)

```
Env: $stream = [\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_{T-1}]$, $\mathcal{M}_0 = \{\}$
Input: experience $t$, global gain $\mathbf{g}_t^r$, global bias $\mathbf{b}_t^r$, learning rate $\eta$, epochs $E$, boolean $update$
$\mathcal{B}_t \leftarrow$ split data $\mathcal{D}_t \cup \mathcal{M}_t$ into a set of batches of size $B$
for epoch $e \in \{0, 1, \dots, E-1\}$ do
    for batch $b \in \mathcal{B}$ do
        Compute the average $\Delta\mathbf{g}^b$, $\Delta\mathbf{b}^b$ over $b$
        $\mathbf{g}_t$, $\mathbf{b}_t \leftarrow \mathbf{g}_t + \Delta\mathbf{g}^b$, $\mathbf{b}_t + \Delta\mathbf{b}^b$
    end for
end for
if update then
    $\mathcal{M}_{t+1} \leftarrow$ UpdateWithStrategy($\mathcal{D}_t$, $\mathcal{M}_t$)
end if
return $\{\mathbf{g}_t, \mathbf{b}_t\}$
```

organized into learning experiences, and each experience unfolds over R communication rounds between the participants. At experience t and round r, the server broadcasts the global parameters θ_t^r to all the clients (line 7). As in `FedIP`, this initializes the clients' side of the algorithm (summarized in Algorithm 4): each client performs E epochs of `IP` on the dataset $\mathcal{D}_t$ and buffer $\mathcal{M}_t$ according to the chosen CL strategy. Then, the client sends the locally updated parameters back to the server (line 8), which updates the local model by aggregating all the received local models (line 10). When the $(R-1)$-th round ends, the server requests all the clients to update their own local buffer to prepare for the next learning experience (condition $r == R-1$ on line 8).

While in this paper we employed a replay strategy based on reservoir sampling ((Vitter, 1985), detailed in Appendix A.3), we believe that this algorithm can be easily extended to any replay strategy (Robins, 1995; Hayes et al., 2019; Shin et al., 2017). Also, with few expedients, it may be extended to *regularization* strategies (Kirkpatrick et al., 2017; Li & Hoiem, 2018; Zenke et al., 2017), which add a regularization term to the loss function of the current experience to consolidate previous knowledge; and *architectural* strategies (Lomonaco & Maltoni, 2017; Mallya & Lazebnik, 2018; Rusu et al., 2016), which aim to isolate model parameters, depending on the task at hand.

3. Experimental Assessment

The objective of the experimental evaluation is to assess the behavior of the proposed IP extensions in each of the corresponding settings. To do so, we designed an experimental setup where we incrementally added complexity to the scenario, starting from a baseline setup in which we assume that all the data is available in advance on a single machine, and ending with the full federated and continual integration. In all settings, we repeated the experiments with different percentages of training clients and assessed: (1) the capability of each algorithm to fit the empirical distribution of the activations to the desired Gaussian distribution; (2) the classification performance of a readout trained on top of a reservoir trained with the proposed algorithms.

The codebase of the experimental assessment is based on Ray (Moritz et al., 2017; Liaw et al., 2018), FedRay[2] and Torch-ESN[3]. In particular, FedRay is a research framework for easy implementation, deployment and scalability of FL experiments. Torch-ESN is a library implementing functionalities and wrappers for ESNs. We make the code publicly available for the sake of reproducibility[4].

3.1. Setup

We tested the `FedIP` and `FedCLIP` on the WEarable Stress and Affect Detection (WESAD) (Schmidt et al., 2018) and the Heterogeneity Human Activity Recognition (HHAR) (Stisen et al., 2015) datasets, two Human Activity Recognition benchmarks. We provide the details of the dataset and their description in Appendix B.1 Both datasets lend themselves to adaptation to federated and continual scenarios. In particular, we split the data into user-specific chunks to simulate the distribution across the devices (15 for WESAD, 9 for HHAR). Then, we simulate drifts by splitting each user-specific chunk into learning experiences dependent on the activity performed by the user in WESAD, and the device worn by the user in HHAR.

As summarized by Figure 2, we built a set of experiments where we incrementally add levels of complexity to the scenario at hand. In the *stationary setting*, we developed a *centralized* baseline, where all the data is available in

[2]https://github.com/vdecaro/fedray/
[3]https://github.com/vdecaro/torch-esn/
[4]https://github.com/vdecaro/federated-esn/tree/exp/neucom

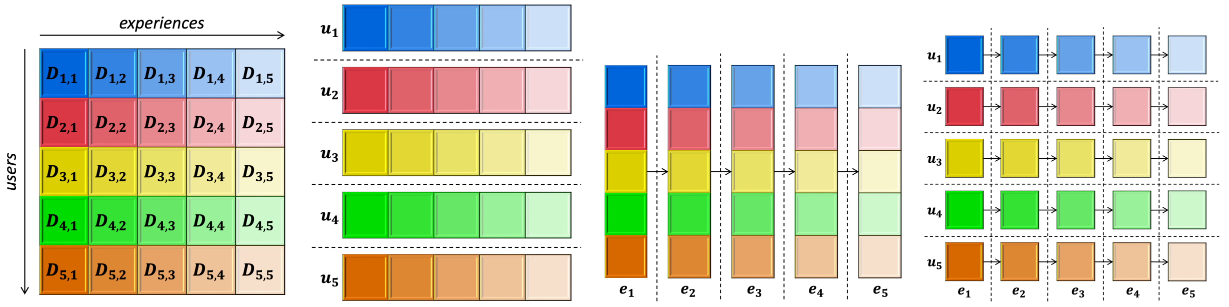
experiences
users
$D_{1,1}$ $D_{1,2}$ $D_{1,3}$ $D_{1,4}$ $D_{1,5}$
$D_{2,1}$ $D_{2,2}$ $D_{2,3}$ $D_{2,4}$ $D_{2,5}$
$D_{3,1}$ $D_{3,2}$ $D_{3,3}$ $D_{3,4}$ $D_{3,5}$
$D_{4,1}$ $D_{4,2}$ $D_{4,3}$ $D_{4,4}$ $D_{4,5}$
$D_{5,1}$ $D_{5,2}$ $D_{5,3}$ $D_{5,4}$ $D_{5,5}$
u_1 u_2 u_3 u_4 u_5
e_1 e_2 e_3 e_4 e_5

Figure 2. Representation of the different data grouping among the four assessed scenarios. Each chunk $D_{i,e}$ represents the data of the i-th client from the e-th learning experience. From left to right. **Full dataset**: a single machine holds all the data in advance. **Federated scenario**: each user holds the full private dataset in advance. **Continual scenario**: a single machine gets the data from clients in a streaming fashion. **Federated and continual scenario**: each user has its own private data stream.

advance on a single machine (Figure 2, left). Here, we assessed the behavior of an ESN trained via `RR` only and via `IP`+`RR`. We extended this baseline towards spatial distribution by experimenting on a *federated and stationary* scenario, where each client has its own, private data as a full dataset available in advance (Figure 2, center-left). Here, we evaluated the performance of an ESN trained via `FedIP` and Federated Ridge Regression (`FedRR`) and compared it with a baseline ESN trained only via `FedRR`. In the *continual setting*, we followed the same approach as in the stationary one. We provided a *centralized* baseline, where the data arrives in a streaming fashion on a single machine (Figure 2, center-right). First, we assessed the behavior of ContinuaL Intrinsic Plasticity (`CLIP`) with two baselines CL strategies: *naïve*, trains both the reservoir and the readout only on the data available from the current experience; *joint*, accumulates all the data up to the current experience and re-trains the model from scratch. Then, we assessed `CLIP` with the Replay strategy with a fixed buffer updated via Reservoir Sampling, where we trained the reservoir as described in Algorithm 7, and the readout by applying `RR` to the union of the data from the current experience and the data available from the buffer. Finally, we assess the behavior of `FedCLIP` in the *federated and continual* setting, where each client has its own, private data stream (Figure 2, right). Here, we applied the same strategies as in the centralized one, i.e., *naïve*, *replay* and *joint*.

In each scenario, we repeated the experiments with four percentages of training clients, i.e., $\{25\%, 50\%, 75\%, 100\%\}$ to assess the generalization capabilities of the algorithms. The workflow of each experiment is described in Appendix B.2. We employed the accuracy to evaluate the performance in the stationary setting, and the stream accuracy in the non-stationary setting. On the WESAD dataset, during the risk assessment phase, we also computed the reservoir's activation density to investigate the behavior of the adapted reservoir. Finally, on all the settings, we verify the results statistically by applying a two-sided T-Test, comparing the performances of the baselines (summarized in Appendix B.3) with the ones of the proposed methods. We consider the differences between the results statistically significant for p-values ≤ 0.05.

3.2. Results

Federated and Stationary Setting In Table 1 we report the test accuracy in the federated and stationary. From these results and Figure 3, we can observe two distinct behaviours of `FedIP`, depending on the percentage of training clients involved. For lower percentages of training clients, i.e., 25% and 50%, we can observe that the ESNs trained with `FedIP` significantly outperform those trained via `FedRR`, with a gain of at least 5 accuracy points (except for HHAR with 50% training clients). In these cases, `FedIP` acts mainly as a regularizer: since the information to be represented with 25% and 50% of clients is low, the algorithm clusters it within Gaussians with small standard deviations (i.e., $\sigma = 0.05$ for 25% of training clients, and $\sigma = 0.1$ for 50%). Instead, for higher percentages of training clients, i.e., 75% and 100%, the performance gain becomes less significant, but still in favor of the ESNs trained via `FedIP`. In this case, `FedIP` maximizes the information gain by dampening the effect of the band-pass filtering applied by the tanh activation. These points suggest that the information gained by the use of `FedIP` improves the generalization capabilities of the ESNs.

Consistently with what we discussed in Section 2.1, training the reservoir via `FedIP` mitigates the variance in the performance in comparison with the ESNs adapted only via `FedRR`. The rationale about the dynamics of an untrained reservoir still holds: initializing the reservoir naïvely does not allow to appropriately represent features that are useful for discriminating the correct label. Instead, adapting the reservoir via `FedIP` allows obtaining good representations of the information even in the face of bad initializations. However, the results on HHAR with 50% of training clients

Table 1. Results of the experiments on the stationary settings. The best results are highlighted in bold, and the results whose difference is not statistically significant (i.e., p-value > 0.05) are highlighted in italics.

%TR	WESAD		HHAR	
	`w/o FedIP`	`w/ FedIP`	`w/o FedIP`	`w/ FedIP`
25%	72.09 ± 0.59	**78.68 ± 0.12**	57.08 ± 3.11	**69.83 ± 0.64**
50%	72.04 ± 1.03	**77.43 ± 0.19**	***63.88 ± 6.02***	***57.74 ± 0.19***
75%	76.53 ± 1.08	**77.97 ± 0.41**	***71.09 ± 0.56***	***71.08 ± 0.69***
100%	77.78 ± 0.58	**79.42 ± 0.39**	*70.29 ± 0.99*	***71.38 ± 0.43***

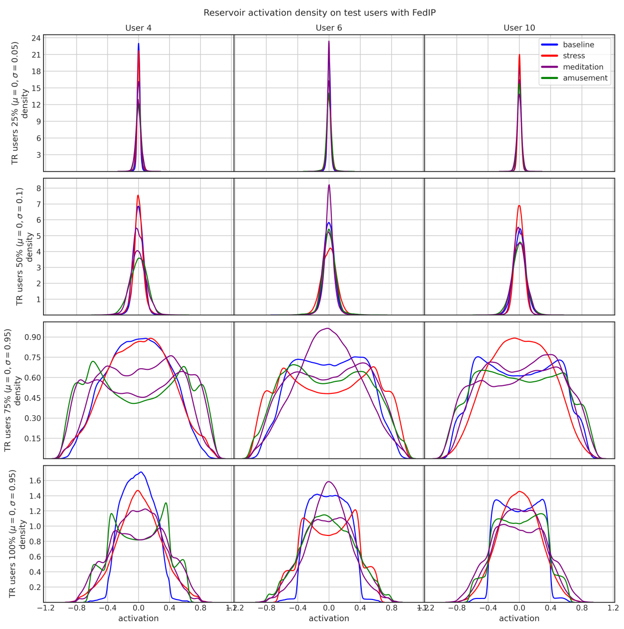

Figure 3. Activation density computed on test clients (columns) with reservoirs trained on different percentages of training clients (rows).

expose a drawback of this effect. Depending on the quality of the input information, the algorithm may filter out also “lucky” initializations. Finally, we can observe that the performance obtained in the federated setting is comparable, if not greater than the one reported for the centralized setting (reported in Table 4). This highlights that not only `FedIP` does not suffer from the approximation given by the model averaging, but it may take advantage of it. As we mentioned, `IP` optimizes the neurons’ parameters to let the activations’ densities converge towards a Gaussian distribution. It is known that a convex combination of n Gaussian distributions is itself a Gaussian distribution. Intuitively, while each client converges to the optimal set of parameters to best fit the Gaussian with respect to the local activations, the aggregation may leverage such property to compose accurate information about the local distributions without suffering from approximations.

Federated and Continual Setting Table 2 reports the stream accuracy of an ESN trained in the federated and non-stationary setting.

On a general note, Table 2 and Figure 4 show that, while the naïve strategy is not able to retain information from previous experiences, and it is prone to forgetting, the joint strategy acts as an upper bound with respect to the performance of the CL strategies. Furthermore, the performances in the federated setting are consistent with the ones of the centralized baseline (reported in Table 5).

On WESAD, the replay strategy is able to achieve the same performance as the joint strategy over the stream with any percentage of training clients. This highlights not only robustness to forgetting, but also the capability of `FedCLIP` to learn the same information as in the stationary scenario with less amount of data. Furthermore, Figure 5 highlights two points. First, we can observe that the distribution of activations gradually adjusts as we proceed through the learning experiences, converging to approximately the same distribution obtained in the stationary case (Figure 3, left). Moreover, paying attention to the distribution of “amusement” activations in Figure 5 (right), we notice that convergence toward its final distribution begins before meeting the data from the corresponding learning experience. This denotes that `FedCLIP` is characterized by good forward transfer in the adaptation of reservoir dynamics.

On HHAR, all the strategies are able to maintain a stable accuracy during the first three learning experiences. This happens because the devices corresponding to these experiences are the three smartphones kept by the user performing the activities. Instead, in the fourth experience, corresponding to the smartwatch, the relation between the movement of the user and the performed activity changes, causing an abrupt concept drift and a consequent decay in performance.

4. Conclusions

In this paper, we have proposed a framework for localized learning based on homeostatic plasticity of dynamical neural systems, based on Reservoir Computing (RC). We extended Intrinsic Plasticity (`IP`), a method to adapt ESNs reservoir dynamics to the input sequence, to a client-server, pervasive scenario with federated and continual data. We proposed

Table 2. Results of the experiments on the federated and continual setting. In the federated setting, the best results are highlighted in bold, and the results whose difference is not statistically significant (i.e., p-value > 0.05) are highlighted in italics.

%TR	WESAD			HHAR		
	Naïve	Replay	Joint	Naïve	Replay	Joint
25%	27.32 ± 10.86	***79.23 ± 0.44***	*78.75 ± 0.67*	34.85 ± 3.08	51.16 ± 5.88	**69.44 ± 0.38**
50%	30.60 ± 7.51	**77.49 ± 0.89**	75.95 ± 1.07	30.16 ± 2.10	43.77 ± 1.50	**60.85 ± 4.37**
75%	51.50 ± 4.10	77.04 ± 0.89	**78.17 ± 0.54**	28.62 ± 0.93	59.83 ± 0.88	**71.14 ± 0.84**
100%	50.80 ± 1.50	77.46 ± 1.31	**79.51 ± 0.35**	30.30 ± 0.43	62.28 ± 0.54	**71.22 ± 0.32**

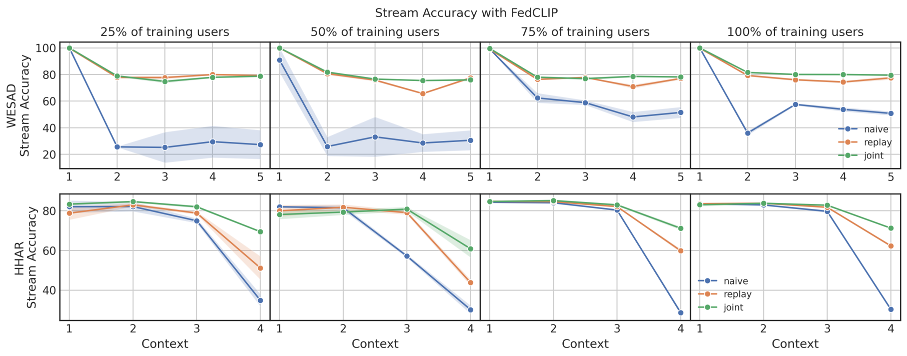

Figure 4. Stream accuracy of an ESN trained via `FedCLIP` as learning experiences progress, with different percentages of training clients.

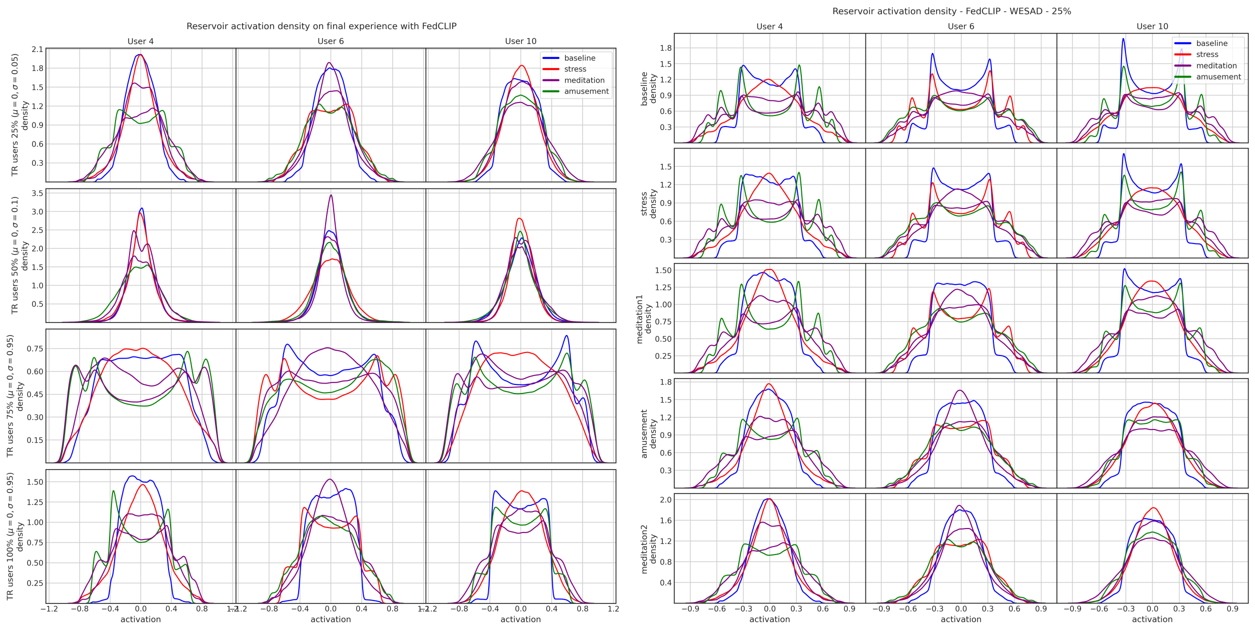

Figure 5. Activation density on the test clients of the global reservoir trained via `FedCLIP`. **Left**: density on the last learning experience with each percentage of training clients. **Right**: density after each learning experience with the 25% of training clients.

`FedIP` for federated and stationary data, and `FedCLIP` for federated and non-stationary data. We tested our algorithms on two HAR benchmarks with incremental setup and different numbers of training clients. The achieved results indicate that our proposals improve the global model performance, achieving comparable results to the joint baseline and their centralized versions, at the same time showing robustness against model averaging approximation.

Acknowledgements

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

References

Bacciu, D., Akarmazyan, S., Armengaud, E., Bacco, M., Bravos, G., Calandra, C., Carlini, E., Carta, A., Cassarà, P., Coppola, M., Davalas, C., Dazzi, P., Degennaro, M. C., Di Sarli, D., Dobaj, J., Gallicchio, C., Girbal, S., Gotta, A., Groppo, R., Lomonaco, V., Macher, G., Mazzei, D., Mencagli, G., Michail, D., Micheli, A., Peroglio, R., Petroni, S., Potenza, R., Pourdanesh, F., Sardianos, C., Tserpes, K., Tagliabó, F., Valtl, J., Varlamis, I., and Veledar, O. Teaching - trustworthy autonomous cyber-physical applications through human-centred intelligence. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pp. 1–6, 2021a. doi: 10.1109/COINS51742.2021.9524099.

Bacciu, D., Di Sarli, D., Gallicchio, C., Micheli, A., and Puccinelli, N. Benchmarking reservoir and recurrent neural networks for human state and activity recognition. In *Advances in Computational Intelligence: 16th International Work-Conference on Artificial Neural Networks, IWANN 2021, Virtual Event, June 16–18, 2021, Proceedings, Part II*, pp. 168–179. Springer, 2021b.

Can, Y. S. and Ersoy, C. Privacy-preserving federated deep learning for wearable iot-based biomedical monitoring. *ACM Transactions on Internet Technology (TOIT)*, 21(1): 1–17, 2021.

De Caro, V., Bano, S., Machumilane, A., Gotta, A., Cassarà, P., Carta, A., Semola, R., Sardianos, C., Chronis, C., Varlamis, I., Tserpes, K., Lomonaco, V., Gallicchio, C., and Bacciu, D. Ai-as-a-service toolkit for human-centered intelligence in autonomous driving. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 91–93, 2022. doi: 10.1109/PerComWorkshops53856.2022.9767501.

De Caro, V., Gallicchio, C., and Bacciu, D. Federated adaptation of reservoirs via intrinsic plasticity. In *Proceedings of the 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2022)*, 2022. URL https://arxiv.org/abs/2206.11087,Arxiv.

French, R. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999. ISSN 13646613.

Gallicchio, C. and Micheli, A. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9:337–350, 2017.

Hayes, T. L., Cahill, N. D., and Kanan, C. Memory Efficient Experience Replay for Streaming Learning. *arXiv:1809.05922 [cs, stat]*, February 2019.

Horvitz, E. and Mulligan, D. Data, privacy, and the greater good. *Science*, 349(6245):253–255, 2015. doi: 10.1126/science.aac4520. URL https://www.science.org/doi/abs/10.1126/science.aac4520.

Huang, T., Lin, W., Wu, W., He, L., Li, K., and Zomaya, A. Y. An Efficiency-boosting Client Selection Scheme for Federated Learning with Fairness Guarantee. *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2020. ISSN 1045-9219, 1558-2183, 2161-9883.

Huang, T., Lin, W., Shen, L., Li, K., and Zomaya, A. Y. Stochastic Client Selection for Federated Learning with Volatile Clients. *arXiv:2011.08756 [cs]*, May 2021.

Jaeger, H. 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*, 148, January 2001.

Jaeger, H. and Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3): 335–352, 2007.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, July 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 0027-8424, 1091-6490.

Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2020.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges. *arXiv:1907.00182 [cs]*, November 2019.

Lesort, T., Caccia, M., and Rish, I. Understanding Continual Learning Settings with Data Distribution Drift Analysis. *arXiv:2104.01678 [cs]*, April 2021.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Li, Z. and Hoiem, D. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, December 2018. ISSN 0162-8828, 2160-9292, 1939-3539.

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., and Stoica, I. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33: 2351–2363, 2020.

Lomonaco, V. and Maltoni, D. CORe50: A new dataset and benchmark for continuous object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 17–26. PMLR, November 2017.

Lukoševičius, M. and Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, August 2009. ISSN 15740137.

Mallya, A. and Lazebnik, S. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *arXiv:1711.05769 [cs]*, May 2018.

McCloskey, M. and Cohen, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Elsevier, 1989. ISBN 978-0-12-543324-2.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017.

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Paul, W., Jordan, M. I., and Stoica, I. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017. URL http://arxiv.org/abs/1712.05889.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

Nguyen, D. C., Pham, Q.-V., Pathirana, P. N., Ding, M., Seneviratne, A., Lin, Z., Dobre, O., and Hwang, W.-J. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(3):1–37, 2022.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, May 2019. ISSN 08936080.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive Federated Optimization. *International Conference on Learning Representations*, May 2021.

Robins, A. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146, June 1995. ISSN 0954-0091, 1360-0494.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, September 2016.

Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., and Van Laerhoven, K. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In *Proceedings of the 20th ACM international conference on multimodal interaction*, pp. 400–408, 2018.

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J., and Stroobandt, D. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, March 2008. ISSN 09252312.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 2994–3003. Curran Associates Inc., 2017. ISBN 978-1-5108-6096-4.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pp. 127–140, 2015.

Triesch, J. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *International Conference on Artificial Neural Networks*, pp. 65–70. Springer, 2005.

Vitter, J. S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, March 1985. ISSN 0098-3500, 1557-7295.

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

Wang, J., Tantia, V., Ballas, N., and Rabbat, M. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In *International Conference on Learning Representations*, 2020b.

Yildiz, I. B., Jaeger, H., and Kiebel, S. J. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3987–3995. JMLR.org, 2017.

A. Algorithms

A.1. Federated Averaging

Algorithm 5 Federated Averaging (`FedAvg`)

```
Input: Initial model $\theta_0$; number of rounds $R$; number of local epochs $E$; learning rate $\eta$
for each round $r \in \{0, 1, \dots, R-1\}$ do
    Sample a subset $\mathcal{S}^r$ from clients $\mathcal{C}^r$
    Broadcast $\theta^r$ to all clients $i \in \mathcal{S}^r$
    for client $c \in \mathcal{S}^t$ in parallel do
        $\theta_c^r \leftarrow \textsc{LocalSGD}_c(\theta^r, \eta, E)$
    end for
    Update global model $\theta^{r+1} \leftarrow \sum_{c \in \mathcal{S}^t} \frac{n_c}{n} \theta_c^r$
end for
```

Algorithm 6 LocalSGD (on client c)

```
Input: Global model $\theta^r$; learning rate $\eta$; number of local epochs $E$
Split local dataset $\mathcal{D}_c$ in set of batches $\mathcal{B}$
$\theta_c^r \leftarrow \theta^r$
for epoch $e \in \{0, 1, \dots, E-1\}$ do
    for batch b $\in \mathcal{B}$ do
        $\theta_c^r \leftarrow \theta_c^r - \eta \nabla \mathcal{L}_c(\theta_c^r, b)$
    end for
end for
return $\theta_c^r$
```

At the beginning of the r-th round, a set of clients $\mathcal{C}^r$ is available, and the server broadcasts θ^r to a random subset of participants $\mathcal{S}^r \subset \mathcal{C}^r$ sampled uniformly. Each client $c \in \mathcal{S}^r$ performs E local SGD steps on its local data and sends its updated local model θ_C^r to the server. Then, the server updates its global model with the formula

$$\theta^{r+1} = \sum_{c \in \mathcal{S}^t} \frac{n_c}{n} \theta_c^r, \tag{9}$$

where n_c is the size of the local datasetof the c-th client and n is the sum of the sizes of the local datasets of clients participating to the r-th round. Notice that, in this algorithm, the variable p_c of eq. (5) is implicitly modelled by the sampling of $\mathcal{S}^t$ and the term $\frac{n_c}{n}$, which model client selection across $\mathcal{C}$ and roughly approximates statistical heterogeneity respectively.

A.2. Continual Intrinsic Plasticity

`CLIP` is the centralized version of `FedCLIP`, and is articulated in learning experiences, one for each task in the given data stream. During the t-th experience, it splits the data from the current dataset $\mathcal{D}_t$ and the memory buffer $\mathcal{M}_t$ in a set of mini-batches $\mathcal{B}_t$ (line 4). Then, it performs E training epochs by applying Intrinsic Plasticity on the mini-batches in $\mathcal{B}_t$ (lines 5-10). When the learning phase is complete, it updates the model (line 11) and the memory buffer (line 12) for the learning experience $(t + 1)$.

The policy for updating the memory buffer and sampling the mini-batches (which refer to lines 12 and 4 respectively in Algorithm 7) depends on the CL strategy at hand. In particular, we applied three main strategies:

- *naïve*, the algorithm is not equipped with a memory buffer and the mini-batches are sampled from the dataset of the current experience $\mathcal{D}_t$;
- *replay with reservoir sampling* (Vitter, 1985), where a bounded buffer is kept balanced with data from each of the previous learning experiences, and each mini-batch is injected with data from the buffer sampled uniformly;
- *joint*, the memory keeps all the data from all the learning experiences up to $\mathcal{D}_t$, and the mini-batches are sampled by chunking $\mathcal{D}_t \cup \mathcal{M}_t$.

Algorithm 7 ContinualIP

```
**Input**: $stream = [\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_{T-1}]$, learning rate $\eta$, epochs $E$
$\theta_0 = \{\mathbf{1}, \mathbf{0}\}$
$\mathcal{M}_0 = \{\}$
**for** $\mathcal{D}_t \in stream$ **do**
    $\mathcal{B}_t \leftarrow$ split data $\mathcal{D}_t \cup \mathcal{M}_t$ into a set of batches of size $B$
    **for** epoch $e \in \{0, 1, \dots, E-1\}$ **do**
        **for** batch $b \in \mathcal{B}$ **do**
            Compute the average $\Delta \mathbf{g}^b,\ \Delta \mathbf{b}^b$ over $b$
            $\mathbf{g}_t,\ \mathbf{b}_t \leftarrow \mathbf{g}_t + \Delta \mathbf{g}^b,\ \mathbf{b}_t + \Delta \mathbf{b}^b$
        **end for**
    **end for**
    $\theta_{t+1} \leftarrow \{\mathbf{g}_t, \mathbf{b}_t\}$
    $\mathcal{M}_{t+1} \leftarrow$ UpdateWithStrategy($\mathcal{D}_t$, $\mathcal{M}_t$)
**end for**
**return** $\theta_T$
```

While the former and the latter represent a lower and upper bound on the performance respectively (Lesort et al., 2019), the replay strategy represents our CL strategy of choice in the proposed setting.

A.3. Replay with Reservoir Sampling

Given a buffer of samples $\mathcal{M}$ of size n, the state of the buffer at experience $i > 0$ is

$$\mathcal{M}_i = \bigcup_{i \in \{0, \dots i-1\}} \mathcal{B}_i,$$

where $\mathcal{B}_i \subseteq \mathcal{D}_i^{train}$ and $|\mathcal{B}_i| = \left\lfloor \frac{|\mathcal{D}_i^{train}|}{\sum_{j \in \{0,\dots i-1\}} |\mathcal{D}_j^{train}|} * n \right\rfloor$. Informally speaking, at experience e_i, the buffer keeps data from all the previous learning experiences while maintaining the same proportions with respect to the sizes of the training datasets. During the learning experience e_i, the mini-batches are injected with samples from both $\mathcal{D}_i^{train}$ and $\mathcal{M}_i$.

B. Further details on experiments

B.1. Datasets Description and Preprocessing

WESAD is a dataset for stress and affect detection from wearable devices. It was collected from 15 participants in a ∼36-minute session where they performed activities depending on the cognitive state to be induced. In particular, data collection unfolded over five main contexts: baseline condition; stress induction; meditation; amusement induction; meditation. Each sample in the resulting time series is equipped with a label corresponding to the expected cognitive states of the user. In our setup, we used a subset of the available data, which consisted of 8 synchronized time series of physiological data sampled at 700Hz by a chest-worn device. We normalized the data of each user and chunked it in non-overlapping sequences of 700 samples (i.e., 1 second).

HHAR is a dataset for activity recognition. It was collected from 9 users keeping 12 smart devices while performing different activities (biking, sitting, standing, walking, stair up, and stair down), to show the heterogeneity of the sensing across the devices. For each user, we selected a subset of samples corresponding to the smartphones LG Nexus4, Samsung Galaxy S3, Samsung Galaxy S3 Mini, and the smartwatch LG Watch. Each sample had 6 features corresponding to the axes of the device's accelerometer and gyroscope, and a label denoting one of the 6 activities performed by the user. For each user and device, we downsampled the sequence to 100Hz to obtain homogeneity of sampling rate across the devices, normalized it, and split the corresponding chunk into non-overlapping sequences of 200 samples (∼2 seconds).

B.2. Hyperparameters and experiment workflow

Table 3. Search space for the two datasets. **Above**: hyperparameters tested on the *stationary* setting. The space spanned by Reservoir and `RR` / `FedRR` is common to **all** the algorithms. The subspaces spanned by `IP` and `FedIP` are employed only in experiments with the corresponding algorithms. . **Below**: hyperparameters tested on the *continual* settings. We constrained the hyperparameter space by using from the best configuration selected on the corresponding centralized and federated settings. Then, we limited this phase to a grid search for selecting the optimal number of learning iterations (i.e., epochs in `IP` and rounds in `FedIP`) to perform in each learning experience.

		WESAD	HHAR
Reservoir	Units	{200,300,400}	$\{100, 200, 300, 400, 500\}$
	$\rho(\hat{\mathbf{W}})$	$[0.3, 0.99)$	$[0.3, 0.99)$
	Input Scaling	$[0.5, 1)$	$[0.5, 1)$
	Leaking Rate	$[0.1, 0.8]$	$[0.1, 0.5]$
`RR` / `FedRR`	L2	$[1e^{-4}, 1]$	$[1e^{-4}, 1]$
`IP`	μ	0	0
	σ	$(0.05, 1)$	$(0.05.1)$
	η	0.01	0.01
	Epochs	$\{10, 12, \ldots, 20\}$	$\{10, 12, \ldots, 20\}$
`FedIP`	μ	0	0
	σ	$(0.05, 1)$	$(0.05.1)$
	η	0.01	0.01
	Global Rounds	$\{10, 12, \ldots, 20\}$	$\{10, 12, \ldots, 20\}$
	Local Epochs	$\{3, 5, 10\}$	$\{3, 5, 10\}$

		WESAD	HHAR
`CLIP`	Exp. Epochs	$ip\_epochs/2 \pm 2$	$ip\_epochs/2 \pm 2$
	Buffer Size	5% full dataset size	5% full dataset size
`FedCLIP`	Exp. Rounds	$fedip\_rounds/5 \pm 2$	$fedip\_rounds/4 \pm 2$
	Buffer Size	5% user dataset size	5% user dataset size

An experiment consisted of three steps:

1. *Model selection*: given the search spaces depicted in Table 3, we performed a random search with 100 configurations if the scenario is stationary, and a grid search if it is continual. We selected the configurations with the highest scores on the data from the validation clients;
2. *Re-training*: given the best configuration selected in step 1, we retrained 5 instances of the model and the corresponding algorithm with the corresponding configuration;
3. *Risk assessment*: we assessed the performance of the 5 instances by computing the metrics on the data from the test clients.

The metrics that we employed for steps (1) and (3) are the accuracy and the stream accuracy for the stationary and continual settings, respectively.

B.3. Centralized Baselines

Table 4. Results on the stationary settings. For each percentage of the users, we report the mean and standard deviation of the test accuracy of each model with and without the use of IP.

%TR	WESAD		HHAR	
	`w/o IP`	`w/ IP`	`w/o IP`	`w/ IP`
25%	72.60 ± 1.24	78.14 ± 0.32	61.34 ± 3.19	68.82 ± 0.49
50%	72.88 ± 1.35	76.98 ± 0.22	58.70 ± 5.29	66.64 ± 2.28
75%	77.06 ± 1.02	78.68 ± 0.38	71.49 ± 0.93	70.33 ± 0.42
100%	79.18 ± 0.40	78.89 ± 0.19	71.71 ± 0.72	70.88 ± 0.74

Table 5. Results on the centralized, non-stationary baseline. We report the mean and standard deviation of the stream accuracy on the last experience for each strategy and percentage of training users.

%TR	WESAD			HHAR		
	`Naïve`	`Replay`	`Joint`	`Naïve`	`Replay`	`Joint`
25%	30.23 ± 1.16	78.37 ± 1.11	78.64 ± 0.90	35.65 ± 4.21	59.56 ± 1.26	69.44 ± 0.37
50%	42.55 ± 8.86	79.91 ± 0.54	76.42 ± 0.44	31.78 ± 0.36	48.01 ± 1.35	68.40 ± 2.53
75%	32.13 ± 6.47	80.27 ± 0.87	79.22 ± 0.33	28.08 ± 0.56	58.93 ± 1.51	70.26 ± 0.49
100%	27.43 ± 0.29	76.19 ± 1.66	79.28 ± 0.52	28.7 ± 0.93	58.52 ± 1.58	70.46 ± 0.26

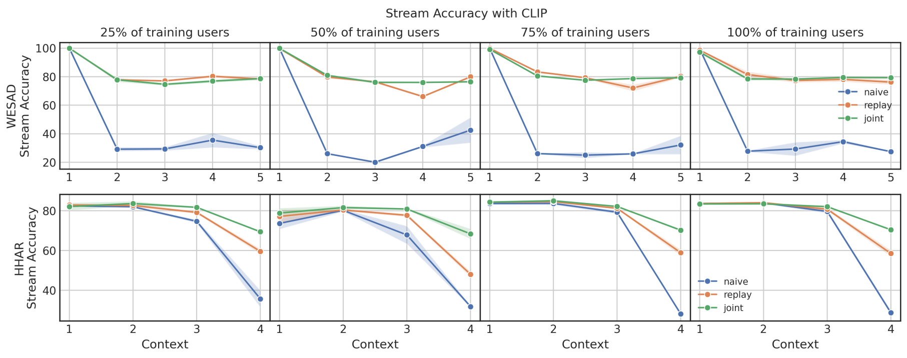

Figure 6. Stream accuracy of the centralized baseline.