hoffset has been altered.

The page layout violates the ICML style.

Please do not change the page layout, or include packages like geometry, savetrees, or fullpage, which change it for you.

We're not able to reliably undo arbitrary changes to the style. Please remove the offending package(s), or layout-changing commands and try again.

Decentralized Plasticity in Reservoir Dynamical Networks for Pervasive Environments

Anonymous Authors[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 to 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 Machine Learning 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 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 that we consider in this paper.

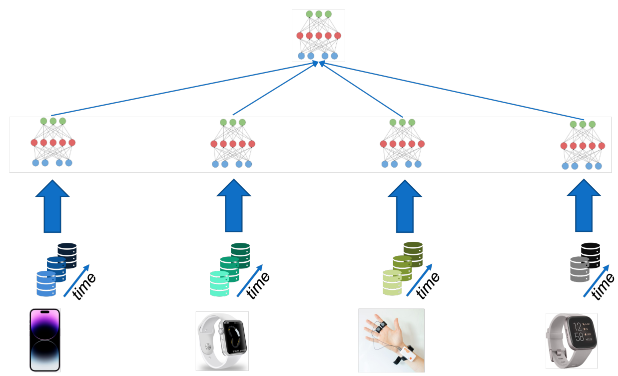

Figure 1. Learning in a pervasive environment

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]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Local dynamics adaptation in pervasive environments

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 $t \in \{0, \dots, T-1\}$, 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.

IP (Triesch, 2005; Schrauwen et al., 2008) is an algorithm inspired by a biological phenomenon, called *homeostatic plasticity*, for adapting the reservoir in an unsupervised manner. From a computational perspective, IP maximizes the entropy of the units' activations. 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. When the non-linearity f is the tanh function, IP minimizes the following Kullback-Leibler divergence:

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

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. The derivation of the loss function leads to the following update rules for the set of gain and bias parameters:

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

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. These simple learning rules allow for maximizing the information content of reservoir states, and to reduce the variance in performance due to random initialization.

In this paper, we aim to extend the use of IP towards the adaptation of the local dynamics in pervasive environment. As we mentioned, the constraints of such an environment impose that (1) devices must collaborate in the learning process without disclosing the data and (2) they must be able to update continually in order to avoid data oblivion.
To address the objective (1), we must reformulate the loss of IP to account for the distribution of the data across clients in the federated scenario. By employing the derivation in Appendix A, the loss function of IP translates to the federated setting as:

$$\mathcal{L}_F(\theta; \mu, \sigma) = \textstyle\sum_{c \in \mathcal{C}} p_c D_{KL}(\tilde{q}_c\,||\,\mathcal{N}_{\mu,\sigma}), \tag{5}$$

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. Our proposal is intended for a client-server topology and is based on Federated Averaging (FedAvg). The algorithm, namely FedIP, instantiates FedAvg 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 is summarized in Algorithm 1 and 3 for the server side and for the client side, respectively. In FedIP, the number of parameters exchanged between a client is $\mathcal{N}_{\mathcal{R}}$.
To address objectives (1) and (2), we must extend our proposal to a continual setting, where all the clients consider

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

Algorithm 1 Federated Intrinsic Plasticity (FedIP)

Input: clients $\mathcal{C}$, number of rounds R, learning rate η, 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 θ^r to client c $\theta^r_c \leftarrow$ IPUpdate$_c$($\mathbf{g}^r$, $\mathbf{b}^r$, η, E) ; // Alg. 3
  end
  $\theta^{r+1} \leftarrow \left\{\sum_{c\in\mathcal{C}} \frac{n_c}{n}\, \mathbf{g}^{r+1}_c,\ \sum_{c\in\mathcal{C}} \frac{n_c}{n} \mathbf{b}^{r+1}_c\right\}$
end

Algorithm 2 Federated Continual Intrinsic Plasticity (FedCLIP)

Input: clients $\mathcal{C}$, learning rate η, 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 $t \in \{0, 1, \ldots, T-1\}$ **do**
  for each round $r \in \{0, 1, \ldots, R-1\}$ **do**
    for each client $c \in \mathcal{C}$ **in parallel do**
      Send θ^r_t to client c $\theta^r_{t,c} \leftarrow$ ContinualIPUpdate$_c$(t, $\mathbf{g}^r_t$, $\mathbf{b}^r_t$, η, E, $r == R-1$)
    end
    $\theta^{r+1}_t \leftarrow \left\{\sum_{c\in\mathcal{C}} \frac{n_{t,c}}{n_t}\, \mathbf{g}^r_{t,c},\ \sum_{c\in[C]} \frac{n_{t,c}}{n_t} \mathbf{b}^r_{t,c}\right\}$
  end
end

that the local data is non-stationary and arrives in a streaming fashion. To do so, we solved (2) by itself by deriving the loss for a continual setting in a centralized scenario, and proposing a replay-based algorithm that suits the mentioned setting (Section B.1). Then, we extended Algorithm 1 towards a scenario where all the clients face their own, local continual scenario.

3. Experiments

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. To maximally exercise our algorithms, we employed four versions for both datasets: centralized; continual; federated; federated and continual. Details on the datasets (including pre-processing and data splitting), and on the experimental setup are provided in Section C.

Federated and stationary setting Table 4 shows the test accuracy over five retraining runs for each setting and percentage of the training clients on both benchmarks.

From the results in Table 4 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 Federated Ridge Regression (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. In an untrained reservoir, the initial dynamics provides no guarantee that the net input of each neuron stands in the range [-3, +3]. This leads the activation function to a contracting behavior by squashing inputs outside this range to -1 and +1. Instead, converging to the desired gaussian with $\sigma < 1$ forces the net input to stand within a range where the dynamics of the reservoir are fully exploited, resulting in better representation capabilities. These points suggest that the information gained by the use of FedIP improves the generalization capabilities of the ESNs.

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 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. 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. A property of such distribution is that, given n Gaussian distributions with parameters $(\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2), \ldots, (\mu_n, \sigma_n^2)$, the distribution of the sum is a Gaussian with parameters $\left(\sum_{i=1}^n \mu_i, \sum_{i=1}^n \sigma_i^2\right)$. This property can be straightforwardly extended to convex combinations of Gaussian distributions. Given this premise, the reason behind the improvement in the performance in the federated setting against the centralized can be intuitively addressed to this property. In the federated setting, each client tends to converge to the optimal set of parameters to best fit the Gaussian with respect to the local activations. Then, the aggregation leverages the aforementioned prop-

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** $\pm$ **0.12**	57.08 ± 3.11	**69.83** $\pm$ **0.64**
50%	72.04 ± 1.03	**77.43** $\pm$ **0.19**	*63.88* $\pm$ *6.02*	***57.74*** $\pm$ ***0.19***
75%	76.53 ± 1.08	**77.97** $\pm$ **0.41**	*71.09* $\pm$ *0.56*	***71.08*** $\pm$ ***0.69***
100%	77.78 ± 0.58	**79.42** $\pm$ **0.39**	*70.29* $\pm$ *0.99*	***71.38*** $\pm$ ***0.43***

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*** $\pm$ ***0.44***	*78.75* $\pm$ *0.67*	34.85 ± 3.08	51.16 ± 5.88	**69.44** $\pm$ **0.38**
50%	30.60 ± 7.51	**77.49** $\pm$ **0.89**	75.95 ± 1.07	30.16 ± 2.10	43.77 ± 1.50	**60.85** $\pm$ **4.37**
75%	51.50 ± 4.10	77.04 ± 0.89	**78.17** $\pm$ **0.54**	28.62 ± 0.93	59.83 ± 0.88	**71.14** $\pm$ **0.84**
100%	50.80 ± 1.50	77.46 ± 1.31	**79.51** $\pm$ **0.35**	30.30 ± 0.43	62.28 ± 0.54	**71.22** $\pm$ **0.32**

erty and is able to compose accurate information about the local distributions without suffering from approximations. Instead, in the centralized setting, the learning algorithm is subject to stochasticity due to the shuffling of the dataset, and may not be able to fit the parameters in order to best represent all the local distributions.

Federated and Continual Setting. Table 5 reports the performance of an ESN trained in the continual setting. In particular, we report the stream accuracy (i.e., the accuracy on the cumulative data up to the current experience) on five retraining runs of an ESN trained via ContinuaL Intrinsic Plasticity (`CLIP`) and `FedCLIP` with the naïve and joint strategies as a baseline, and with the proposed replay strategy.

On a general note, Table 5 and Figure 3 show that the two baseline strategies behave as expected. 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.

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 4 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. Moreover, paying attention to the distribution of "amusement" activations 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, we observe a different trend. 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.

Finally, as happens in the stationary scenario, Table 5 highlights that `FedCLIP` is able to achieve performances equal or greater than the centralized baseline.

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 `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.

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.

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

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.

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.

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.

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

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.

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.

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.

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.

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

A. FedIP

A.1. Derivation

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):

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

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:

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

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.

A.2. Client-Side Algorithm

Algorithm 3 IPUpdate (on client c)

Input: global gain $\mathbf{g}^r$, global bias $\mathbf{b}^r$, learning rate η, local epochs E

```
g_c, b_c ← g^r, b^r  Split local data into a set of batches 𝓑 of size B  for epoch e ∈ {0, 1, ..., E − 1} do
    for batch b ∈ 𝓑 do
        Compute the average Δg^b, Δb^b over b ;                    // Eq. (4)
        g_c, b_c ← g_c + Δg^b, b_c + Δb^b
    end
end
return g_c, b_c
```

B. Federated Continual Intrinsic Plasticity

B.1. Continual Intrinsic Plasticity

In the continual setting, we rely on the same formalization by Lesort et al. (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:

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

Algorithm 4 ContinualIP

Input: $stream = [\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{T-1}]$, learning rate η, epochs E

22 $\theta_0 = \{\mathbf{1}, \mathbf{0}\}$ $\mathcal{M}_0 = \{\}$; `// Memory buffer, initially empty`

23 **for** $\mathcal{D}_t \in stream$ **do**

24     $\mathcal{B}_t \leftarrow$ split data $\mathcal{D}_t \cup \mathcal{M}_t$ into a set of batches of size B **for** epoch $e \in \{0, 1, \ldots, E-1\}$ **do**

25   **for** batch $b \in \mathcal{B}$ **do**

26   Compute the average $\Delta \mathbf{g}^b$, $\Delta \mathbf{b}^b$ over b ; `// Eq. (4)`

27             $\mathbf{g}_t, \mathbf{b}_t \leftarrow \mathbf{g}_t + \Delta \mathbf{g}^b, \mathbf{b}_t + \Delta \mathbf{b}^b$

28   **end**

29   **end**

30     $\theta_{t+1} \leftarrow \{\mathbf{g}_t, \mathbf{b}_t\}$ $\mathcal{M}_{t+1} \leftarrow$ UpdateWithStrategy($\mathcal{D}_t$, $\mathcal{M}_t$)

31 **end**

32 **return** θ_T

However, in the continual setting, the realizations of the context variable $p_{k,t}$ are not available, 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.

To address this problem, we propose `CLIP`, which extends `IP` to cope with NI scenarios with known task boundaries. As described in algorithm 4, `CLIP` 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 4) 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 $\mathcal{D}_t \cup \mathcal{M}_t$.

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.

B.2. Client-Side Algorithm

Algorithm 5 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 η, 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 ; `// Eq. (4)`
    $\mathbf{g}_t,\ \mathbf{b}_t \leftarrow \mathbf{g}_t + \Delta \mathbf{g}^b,\ \mathbf{b}_t + \Delta \mathbf{b}^b$
  end
end
if $update$ **then**
  $\mathcal{M}_{t+1} \leftarrow$ UpdateWithStrategy($\mathcal{D}_t$, $\mathcal{M}_t$)
end
return $\{\mathbf{g}_t, \mathbf{b}_t\}$

C. Experimental setup

Datasets Description and Pre-processing WESAD is a dataset for stress and affect detection from wearable devices. It was collected from 15 participants in a $\sim$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).

The second dataset is HHAR, which 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 ($\sim$2 seconds).

Data splitting First, we performed a user-wide split into training, validation, and test sets on both datasets. Given the user-specific chunks (15 for WESAD, 9 for HHAR), we applied a 9-3-3 and 5-2-2 split for WESAD and HHAR respectively. With these splits, we were able to simulate the local private data of the clients. In the federated setting, each split fulfills a particular purpose: the training users are involved in the learning process; the validation users monitor the performance of the trained models for model selection; the test users assess the performance of clients joining the federation after training is over. Then, we split 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. In the continual setting, this split allowed us to simulate a continuous data stream that exposes domain drifts over time. The resulting representations are depicted in Figure 2.

In the stationary setting, we developed a *centralized* baseline, where all the data is available in advance on a single machine. 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. Here, we evaluated the performance of an ESN trained via `FedIP` and `FedRR` and compared it with a baseline ESN trained only via `FedRR`.

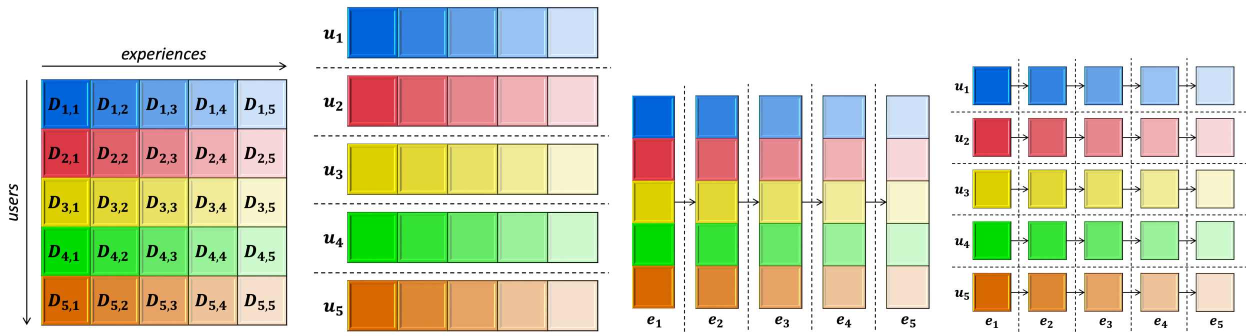

Figure 2. Representation of the different data grouping among the four assessed scenarios. Each chunk $D_{i,e}$ represents the data of the i-th client from the e-th learning experience. From left to right. **Full dataset**: a single machine holds all the data in advance. **Federated scenario**: each 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.

Table 3. Search space for the two datasets on the stationary and continual settings.

((a)) Hyperparameters tested on the **stationary** settings. The search 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.

		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	$\{3, 5, 10\}$	$\{3, 5, 10\}$

((b)) 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
`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

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. First, we assessed the behavior of `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 4, 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. Here, we applied the same strategies as in the centralized one, i.e., *naïve*, *replay* and *joint*.

Table 5. Results of the experiments on the centralized, continual setting. 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

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. Given one of the four scenarios and a percentage of training clients, the corresponding 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. The latter is defined as

$$SACC_t = \frac{1}{\sum_{i=0}^{t} N_i} \sum_{j=0} ACC_j * N_j, \tag{8}$$

where N_j and ACC_j are respectively the number of samples and the accuracy of the model on the data from the j-th learning experience. 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 with the ones of the proposed methods. We consider the differences between the results statistically significant for p-values ≤ 0.05.

D. Experimental Results

D.1. Centralized Baseline Results

Table 4. Results of the experiments on the centralized baseline of the stationary setting. 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

D.2. Results in the Continual Settings

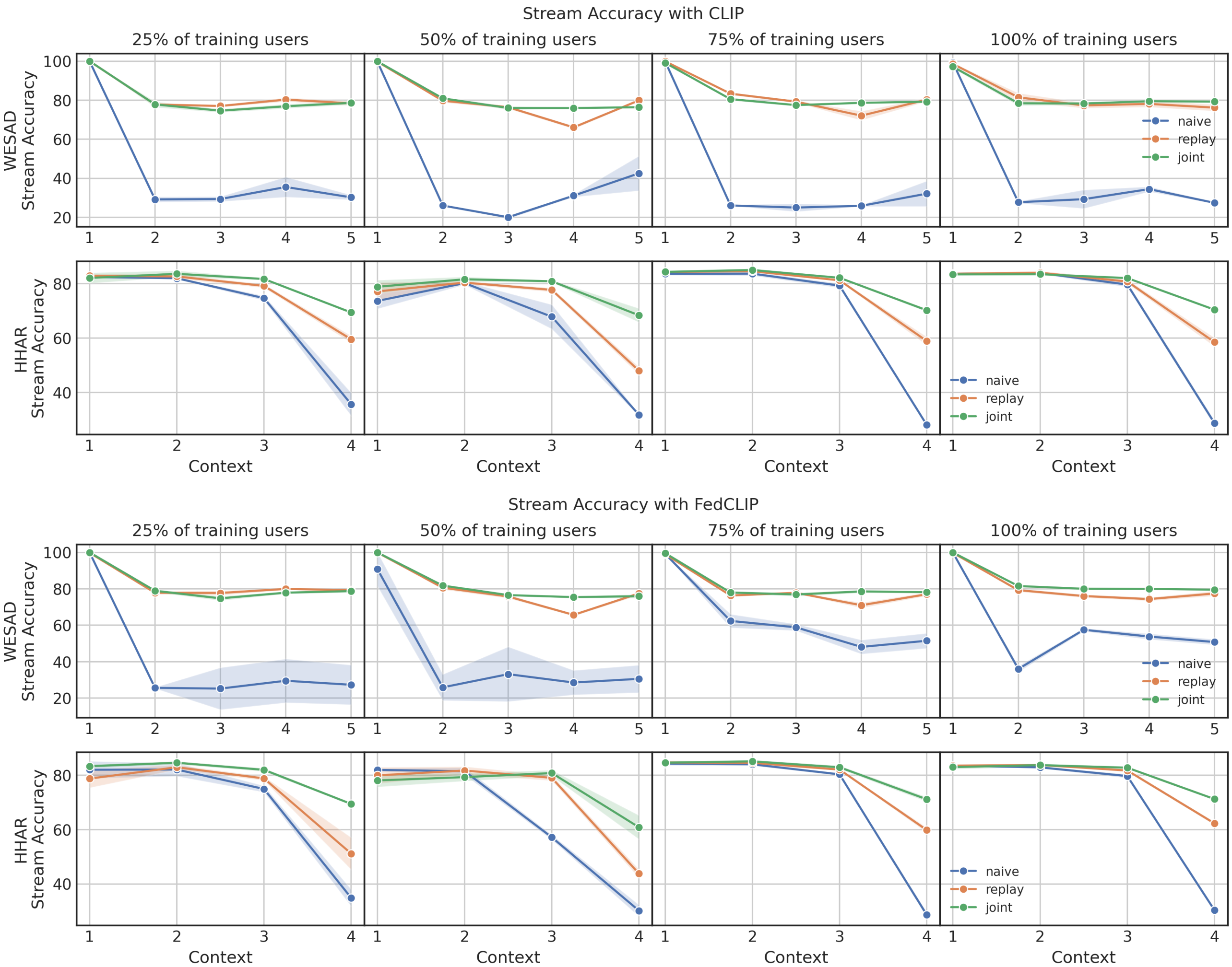

Figure 3. Stream accuracy of an ESN trained via `CLIP` (left) and `FedCLIP` (right) as learning experiences progress, with different percentages of training clients.

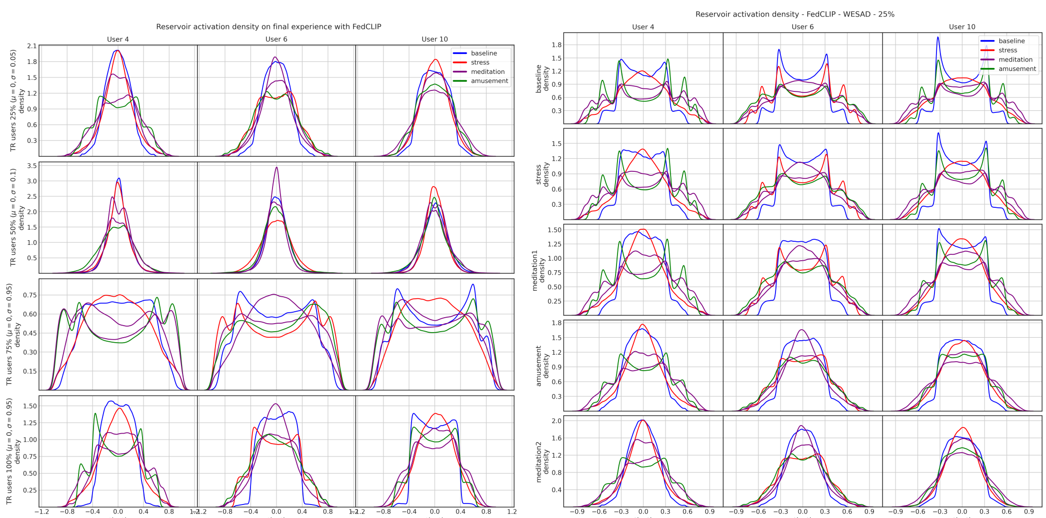

Figure 4. Left: Activation density on the last learning experience with each percentage of training clients. Right: Activation density on test clients (columns) with reservoirs trained via `FedCLIP`.