FedCluLearn: Federated Continual Learning using Stream Micro-Cluster Indexing Scheme *

Milena Angelova[1](✉), Veselka Boeva[1], Shahrooz Abghari[1], Selim Ickin[2], and Xiaoyu Lan[2]

[1] Blekinge Institute of Technology, Karlskrona, Sweden {milena.angelova, veselka.boeva,shahrooz.abghari}@bth.se

[2] Ericsson AB, Stockholm, Sweden {selim.ickin,xiaoyu.lan}@ericsson.com

Abstract. Artificial Neural Networks (NNs) are unable to learn tasks continually using a single model, which leads to forgetting old knowledge, known as *catastrophic forgetting*. This is one of the shortcomings that usually plague intelligent systems based on NN models. Federated Learning (FL) is a decentralized approach to training machine learning models on multiple local clients without exchanging raw data. A paradigm that handles model learning in both settings, federated and continual, is known as Federated Continual Learning (FCL). In this work, we propose a novel FCL algorithm, called FedCluLearn, which uses a stream micro-cluster indexing scheme to deal with catastrophic forgetting. FedCluLearn interprets the federated training process as a stream clustering scenario. It stores statistics, similar to micro-clusters in stream clustering algorithms, about the learned concepts at the server and updates them at each training round to reflect the current local updates of the clients. FedCluLearn uses only active concepts in each training round to build the global model, meaning it temporarily forgets the knowledge that is not relevant to the current situation. In addition, the proposed algorithm is flexible in that it can consider the age of local updates to reflect the greater importance of more recent data. The proposed FCL approach has been benchmarked against three baseline algorithms by evaluating its performance in several control and real-world data experiments.[3]

Keywords: Federated continual learning · Catastrophic forgetting · Concept drift · Data stream clustering · Time series data

1 Introduction

Federated Continual Learning (FCL) is built upon various decentralized devices, such as the Internet of Things (IoT) or smartphones, that constantly produce data to train models. Concept drift is a challenge for these models, as it leads

* This research was funded partly by the Knowledge Foundation, Sweden, through the Human-Centered Intelligent Realities (HINTS) Profile Project (contract 20220068).

[3] The implementation of FedCluLearn and the experimental results are available at https://github.com/milenaangelova1/FedCluLearn.

to unpredictable changes in the statistical characteristics of the data over time. These changes can occur due to factors such as shifting user behavior or external circumstances. As a result, models can quickly become outdated when the data evolves. Therefore, it is crucial to implement methods for detecting and adapting to these changes. Artificial Neural Network (NN) models can suffer from a problem called *catastrophic forgetting*. This problem occurs when the model forgets information that it has learned in the past. This issue arises when the model is updated to work with data that has a different distribution than before. To tackle this problem, the concept of continual learning is introduced. Continual Learning (CL) aims to create models that can learn new information while keeping important insights from earlier data. It uses techniques such as memory replay mechanisms [31, 27, 25] to remind the model of past data and regularization techniques [4, 18] to penalize changes that could harm previous learning. In [23], modular deep learning is considered as a promising solution to the challenges associated with developing models that specialize in multiple tasks.

In this study, we propose a novel FCL approach, called FedCluLearn, inspired by stream clustering algorithms such as CluStream [2] and ClusTree [15], to address the challenges of concept drift and catastrophic forgetting. These algorithms divide the clustering process into an *online* component that uses a micro-clustering approach to periodically store detailed summary statistics and an *offline* component that uses these summary statistics. In the context of FCL, the online component stores summary statistics about the locally learned concepts at each training round, while the offline component uses the stored information at the server to build the global model for that round. As discussed in [2], the efficient storage and use of statistical data for processing evolving data streams is a challenging problem. The authors introduce the concept of a pyramidal time frame combined with a micro-clustering approach. The pyramidal time frame helps determine the optimal moments for storing snapshots of statistical information. The micro-clusters maintain statistical information about the data locality, which defines a temporal extension of the cluster feature vector. FedCluLearn algorithm incorporates the micro-cluster indexing scheme, allowing for the storage of snapshots of the statistical information about local model updates during each global training round. In this study, the proposed FedCluLearn and its variation based on FedProx [17] optimization, called FedCluLearn-Prox, are studied and evaluated on real-world data. In addition, several control experiments are simulated to explore the algorithms' ability to deal with catastrophic forgetting and concept drift. The performance of FedCluLearn and FedCluLearn-Prox is compared to that of FedAvg [20], FedAtt [6], and FedProx [17].

2 Related work

The main concepts in the CL paradigm include continuously acquiring, updating, accumulating, and exploiting knowledge [5, 28]. CL algorithms have to deal with several challenges, such as catastrophic forgetting, data distribution shift, and issues related to imbalanced data and scarcity of labeled data. Catastrophic forgetting [7] refers to model performance degradation due to changes in data distribution cornering, e.g., the appearance of a new concept, which can lead to a downgrade in the model performance over previously learned concepts. Overall, catastrophic forgetting poses a significant challenge in the context of CL, which inherently involves learning incrementally from data. Additionally, there are other important challenges to consider, such as data distribution shifts, also known as concept drift. CL models must effectively handle such phenomena to prevent catastrophic forgetting [16]. The recent survey by Wang et al. [28] grouped CL methods into five major categories 1) regularization- 2) replay-, 3) optimization-, 4) representation-, and 5) architecture-based approaches.

Federated Learning (FL) is a distributed ML approach to train models on multiple local clients without exchanging raw data from client devices to a central server [20]. However, most existing solutions ignore the CL of incremental tasks from streaming data environments. An emerging paradigm that addresses model learning in both federated and continual learning environments is Federated Continual Learning (FCL). FCL combines the strengths of both FL and CL to establish a robust foundation for Edge-AI in dynamic and distributed environments [29]. The study [29] surveys FCL methods considering three federated task characteristics, namely class CL, domain CL, and task CL. While class CL aims to recognize new classes over time, it struggles with task identification, especially when combined with FL. Domain CL addresses the issue of dataset distribution across clients, with each client managing its private dataset as a separate domain. Concerning task CL, local clients learn various tasks and share their knowledge to develop a global model, which helps update and distribute knowledge about diverse tasks among them.

A recent survey on FCL presented in [30] discusses the integration between FL and CL, in particular via knowledge fusion. The study describes that local knowledge can be extracted from three main parts: data, models, and outcomes. Overall, existing knowledge fusion methods are divided into seven classes, where rehearsal and clustering belong to the data category, all gradients/parameters, parameter/layer isolation, and dynamic architecture belong to the models, and prototype and knowledge distillation concern output. The proposed classification considers FedAvg [20], under all gradients/parameters category, and FedProx [17] and FedAtt [6] to dynamic architecture. FedAvg [20] aggregates local model updates using a simple weighted averaging for building the global model. FedProx [17] is a generalization and re-parametrization of FedAvg with the capability to tackle heterogeneity in federated networks belonging to a dynamic architecture. FedAtt [6] builds a personalized FL method by incorporating attention-based grouping to facilitate collaborations among similar clients.

To improve the FL model performance, previous studies such as [10, 21, 11], leverage clustering to group similar clients. While [10, 21] use clustering approaches, they assume a fixed number of client groups throughout the training with no change in clients' data distribution. The study presented in [11] proposes grouping clients into clusters in a one-shot manner by measuring the similarity degrees among clients based on local models' weights. However, none of these works are suitable for the CL framework. The study [13] applies clustering to group the client models with similar concepts while running concept-matching algorithms that collaboratively train and update each model with data relevant to its concept.

According to the classification introduced in [29], the proposed FedCluLearn can be considered mostly related to the federated *domain* CL category, while it fits into the *model* class according to the classification in [30]. Compared with the solutions reviewed above, FedCluLearn proposes a new efficient way to deal with concept drift and catastrophic forgetting by storing and managing statistics about previously learned local models at the server. This allows FedCluLearn to make smart use of the stored information by aggregating into the global model only local updates currently relevant to the situation, and by choosing whether to use more recent or older clients' updates.

Researchers have extensively studied concept drift in data stream mining, proposing various detection and adaptation strategies. According to [9], there are two types of concept drift *virtual* and *real*. While virtual concept drift affects the input data distribution due to, e.g., imbalanced data, real concept drift is caused by, e.g., the appearance of new classes, concepts, or tasks, which can mainly be detected due to the model performance degradation. Agrahari et al. [3] classify drift detection methods into categories such as statistical significance, window-based, and model-dependent techniques. Similarly, Iwashita et al. [12] review approaches based on sliding windows, instance weighting, and classifier ensembles. Early works by Gama et al. [8] and Wadewale et al. [26] also define core types of concept drifts as *sudden*, *gradual*, *incremental*, *recurring*, and *blip or noise* where each of them affecting learning systems differently.

In the context of our FCL setting, we define concept drift at *local* and *global* levels. For example, sudden drift can affect one or more clients and potentially cause the global model's performance to deteriorate during global rounds. Gradual and incremental drifts can lead to inconsistencies among clients and cause the model to converge globally more slowly. Recurring drifts highlight the need for memory-aware strategies at local and global levels, as well as a model that can adapt to changes. Blips and noise require robust aggregation to prevent global degradation. Although the proposed FedCluLearn primarily addresses real concept drift, both virtual and real drifts [9] pose challenges for local and global models during the distributed training process.

3 FedCluLearn algorithm

The main idea of the proposed FCL algorithm, FedCluLearn, is to interpret the FL training process as a stream clustering scenario. The FedCluLearn consists of two main phases: *Initialization* and *Iteration*. The different steps of these phases are described below. In addition, their pseudo codes are presented in Algorithm 1.

I. FedCluLearn Initialization Phase (see Algorithm 1):

I.1. Build the first global NN model by randomly initializing its weights.

I.2. Send the global model to the clients to be trained locally and return the local models' weights to the server.

I.3. Cluster the local models' weights into k clusters (initially learned concepts).

I.4. Compute the statistics of each cluster i, which initially has two feature vectors. One represents the first training round, and the other holds the overall cluster statistics up to the current training round. Note that a new feature vector will be created to store the statistics of each next round. The two types of feature vectors, CF_{ir} $(r = 1, 2, \ldots)$ and $\boldsymbol{CF_{iT}}$, for cluster i are presented in (1). More details about them are given below.

$$\begin{aligned} CF_{ir} &= [\ n_{ir} \quad LS_{ir} \quad SS_{ir} \quad F_{ir}] \\ \boldsymbol{CF_{iT}} &= [\ \boldsymbol{n_{iT}}\ \boldsymbol{LS_{iT}}\ \boldsymbol{SS_{iT}}\ \boldsymbol{F_{iT}}] \end{aligned} \tag{1}$$

- *The feature vector* CF_{ir} created at training round r $(r = 1, 2, \ldots)$ for cluster i contains the following information: (i) number of clients n_{ir}; (ii) linear sum of clients' local models' parameters LS_{ir}; (iii) squared sum of clients' local models' parameters SS_{ir}; (iv) an n dimensional binary vector F_{ir} showing which clients are assigned to this cluster at this training round, where n is the total number of clients.
- *The overall cluster feature vector* $\boldsymbol{CF_{iT}}$ for cluster i can be obtained by $\sum_{r=1} CF_{ir}$. Note that here $\boldsymbol{F_{iT}}$ is an n-dimensional frequency vector showing clients' frequency of being assigned to this cluster.

In addition to the feature vectors, a cluster that receives new items during the current training round is flagged as the one representing an active concept.

I.5. The clusters (learned concepts) are organized in a list of pairs. The pair of cluster i, for $i \in \{1, 2, \ldots, k\}$, at the initialization phase contains two identical cluster feature vectors CF_{i1} and $\boldsymbol{CF_{iT}}$. In general, the first component of the pair stores the training rounds' feature vectors, while the second component contains only the total feature vector.

I.6. The global model G_r for the current round r is computed by averaging the recent linear sums of the active clusters $\Big\{\{LS_{it}\}_{t\in\mathcal{T}}\Big\}_{i\in\mathcal{K}}$, where $\mathcal{K}$ is the set of clusters marked as active in this training round and $\mathcal{T}$ is the set of rounds considered in the global model aggregation. Note that $\mathcal{T}$ can include all training rounds or a percentage of the total training rounds. The expression used to calculate the global model G_r is given in (2).

$$G_r = \sum_{i\in\mathcal{K}} \sum_{t\in\mathcal{T}} LS_{it}/n_{it}. \tag{2}$$

Note that the formula in (2) can be replaced by another FL aggregation scheme. This way, as we show in our experiments, the used micro-cluster indexing scheme can be combined with other FL algorithms.

II. FedCluLearn Iteration Phase (see Algorithm 1):

II.1. At each training round r, for $r = 2, \ldots$, the global model G_{r-1} built at the previous round is sent to the clients to be trained locally, and after that, the local models' weights are returned to the server.

II.2. For each client's weights, the closest cluster is initially found. The Euclidean distance, or any other suitable distance measure, can be used to estimate the similarity between the client's weights and the mean vector of each cluster. The mean vector is calculated by using the cluster's overall feature vector.

II.3. When the nearest cluster is identified, two scenarios are evaluated: either the cluster is a suitable candidate to incorporate the client's updates, or the updates are too distant from this cluster, resulting in the formation of a new singleton cluster. There are different ways to figure out if the client belongs to the cluster, see Section 3.1 for more information.

II.4. The last step is to build the global model by applying (2). Then FedCluLearn returns to step *II.1*. Note that in this step, only active clusters are considered, meaning that FedCluLearn temporarily forgets the knowledge that is not relevant to the current situation.

Algorithm 1 FedCluLearn algorithm

```
1: Input: $R$: number of global rounds
2: Initialize $w_0$
3: for each round $r \in R$ do
4:     Send $w_0$ for local training
5:     if $r == 0$ then                                              ◁ Initialization Phase
6:         $clients \leftarrow$ Send the clients' updates to the global server
7:         $C = \{C_1, C_2, ...C_i\} \leftarrow$ Group the $clients$ in $k$ clusters then
8:         for each $c \in$ C do
9:             $cStats \leftarrow$ Calculate the LS, SS for cluster $c$
10:            $listOfPairs \leftarrow$ Represent each $c \in cStats$ as pair $\langle CF_{cr}, CF_T \rangle$      Eq. (1)
11:        end for
12:    else
13:        for each $c \in listOfPairs$ do                               ◁ Iteration Phase
14:            $maxSI \leftarrow$ Find the SI score between each $client$ and the current $c$
15:            if $maxSI \geq threshold$ then
16:                $listOfPairs \leftarrow$ Update the cluster's statistics for that cluster $c$
17:            else
18:                Create a new cluster $c_{i+1}$ and append it to the $listOfPairs$
19:            end if
20:        end for
21:    end if
22:    Update the $CF_T$ for each cluster $c \in listOfPairs$
23:    Build the $G_r$ based on clusters' statistics from $listOfPairs$         Eq. (2)
24:    Send the $G_r$ to all clients for the next training
25: end for
```

3.1 FedCluLearn design choices

In this section, we discuss the design choices of our FedCluLearn algorithm, first with respect to what information to store in the cluster feature (CF) vectors and when. Second, and related to this, is how to decide whether a client's model update belongs to its nearest micro-cluster centroid.

FedCluLearn stores a list of CF vectors, one at each training round, and updates the statistics of the overall CF vector, see (1). The overall CF vector contains information about: (i) the total number of clients assigned to the cluster;

(ii) the linear sum of clients' model updates; (iii) the squared sum of clients' model updates; and (iv) the frequency with which the clients are assigned to that cluster. Information from (i) to (iv) is also stored in each CF vector kept at each training round and used to update the respective parts of the overall CF vector. Note that (i) and (ii) of the overall CF vector are used to compute the cluster centroid, which is needed to identify the closest cluster for each client's model updates at each training round. In addition, (i) and (ii) of the list of CF vectors stored at the training rounds of active clusters are used to compute the global model, see (2). Note that (iii) of the overall CF vector can be used, similar to the solution in [2], to define the maximum boundary of the cluster and determine whether a client update belongs to it. Finally, (iv) can be used to monitor and analyze clients' behavior during the federated training process, e.g., this can reveal clients with unstable behavior frequently changing their concepts.

In the current study, in step *I.3* k-means [19] is used for the initial grouping of the local models. Other clustering algorithms can also be applied, e.g., affinity propagation and Markov clustering [1]. In addition, in step *II.3* of FedCluLearn, we have used the Silhouette Index (SI) [24] to decide whether to assign a client model update to one of the existing clusters or to create a new cluster. The SI values vary between -1 and 1. FedCluLearn utilizes a predefined threshold to assess when the client's local model updates are similar enough, allowing them to be included in the statistics of an existing cluster. The threshold can be defined empirically before the start of the FL training or dynamically at each round.

4 Experimental setup

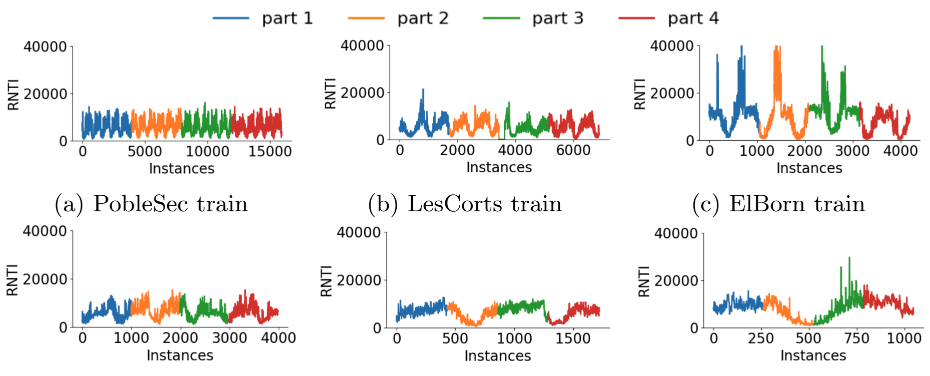
4.1 Data and implementation

We have used two real-world datasets, **5G network dataset** and **Air Quality dataset**, in our experiments.

The first is a **5G network dataset** [22] collected from three distinct urban locations in Barcelona, Spain, representing tourist (El Born), entertainment (Les Corts), and residential (Poble Sec) zones. These sites serve as individual FL nodes. The datasets preserve user anonymity and contain detailed traces of network usage aggregated over two-minute intervals. The data spans the following periods:

- **ElBorn**: 5,421 samples, from 2018-03-28 15:56:00 to 2018-04-04 22:36:00
- **LesCorts**: 8,615 samples, from 2019-01-12 17:12:00 to 2019-01-24 16:20:00
- **PobleSec**: 19,909 samples, from 2018-02-05 23:40:00 to 2018-03-05 15:16:00

Each record contains 11 aggregated features and five target variables, including uplink and downlink traffic volumes (Up, Down), RNTI (Radio Network Temporary Identifier) count, and the number of downlink and uplink resource blocks (RB Down, RB Up). Following the setup in [22], we treat the three nodes as Non-IID (Non-Independent and Identically Distributed) FL clients due to the varying sample sizes and distributional differences across sites. In contrast to previous work [22], which predicted multiple traffic-related features, we focus on the RNTI count as the target variable. The data distributions are depicted in Fig. 1.



(a) PobleSec train (b) LesCorts train (c) ElBorn train

(d) PobleSec test (e) LesCorts test (f) ElBorn test

Fig. 1: 5G data distributions of train and test datasets divided into four equal partitions in experiment $B.1$.

The second dataset is the **Air Quality dataset** [32], which contains hourly measurements of PM2.5 concentrations collected from 12 monitoring stations across Beijing, China, between March 2013 and February 2017. The data originates from the Beijing Municipal Environmental Monitoring Center and is complemented with meteorological information from 15 weather stations operated by the China Meteorological Administration. The meteorological variables include air temperature, wind speed and direction, atmospheric pressure, relative humidity, and precipitation, with measurements provided at six-hour intervals. These additional variables are used better to capture the influence of weather conditions on PM2.5 levels and to enable spatial-temporal adjustments of pollution estimates. In total, each monitoring site provides approximately 35,000 hourly records over the four years. The full dataset contains 18 features, which combine air quality indicators and weather-related attributes. The target variable in this case is PM2.5 concentration, a key indicator of air pollution. Similar to the 5G dataset, we treat the 12 nodes as Non-IID FL clients due to the varying sample sizes and distributional differences.

In our implementation, FedCluLearn initializes an NN with input sizes of 10 (5G network) and 9 (Air Quality), two hidden layers (128 and 64 neurons), leaky ReLU activations, and a scalar output. Only numerical features are used, with missing values in the Air Quality dataset imputed with zero. Models are trained locally for three epochs using a batch size of 128 and a learning rate of 0.0001. Initial client clustering is performed using k-means, with the number of clusters and client assignments guided by the SI. A fixed SI threshold of 0.5, chosen empirically, is used throughout. Both datasets are split 80/20 for training and testing and exhibit non-stationary distributions, making them suitable for evaluating FL robustness under concept drift. Information about pre-processing details and data distributions is available in our public repository.

4.2 Baseline algorithms

Three baseline FL algorithms are used as benchmarks in our experiments. These are **FedAvg**, **FedProx**, and **FedAtt**.

Federated Averaging (FedAvg) [20] is a distributed learning algorithm where multiple clients train local models on their own data without sharing it. A global model is then created by averaging these local models. However, FedAvg struggles with catastrophic forgetting as new updates overwrite past knowledge. Furthermore, it cannot detect and adapt to concept drift, which can lead to a decline in performance over time.

To improve training stability, **Federated Optimization (FedProx)** [17] extends FedAvg by incorporating a proximal term that limits local model updates, ensuring they remain close to the global model. While this helps when clients have heterogeneous data, FedProx still does not explicitly address concept drift, making it vulnerable to performance degradation and catastrophic forgetting as data distributions change.

To tackle concept drift more effectively, **Federated Attention (FedAtt)** [6] enhances FedAvg by incorporating a mechanism to retain past information effectively while adapting to sudden changes in data patterns, thereby minimizing forgetting of older knowledge over time. The used baselines are representative of widely used FL approaches. Despite their advancements, none of these methods are explicitly designed for continual learning or concept drift scenarios. We include them in our evaluation to demonstrate how models lacking explicit mechanisms for long-term memory and adaptation can struggle under evolving data distributions. We do not intend to position FedCluLearn as a direct replacement, but rather to emphasize the importance of incorporating continual learning principles into federated setups.

4.3 Experiments

In this study, we investigate the effectiveness of our FedCluLearn model in mitigating catastrophic forgetting and dealing with concept drift in an FCL setting. The aim is to establish whether the model can retain previously acquired information while adapting to newly emerging concepts and to further analyze its performance in managing data trends over time. We have conducted a series of control experiments alongside experiments using real-world data. The control experiments are designed to explore the impact of concept drift and catastrophic forgetting on the behavior of FedCluLearn under small control scenarios. The experiments with real-world data can provide insight into the performance of the model in a real-world context.

Control experiments are grouped into three categories, representing different learning and concept drift scenarios. We have simulated parallel and continual learning of new concepts, leading to different concept drift scenarios. In the three experimental setups conducted, see Fig. 2, three clients participated in over 200 global rounds. The data represents three distinct concepts. Every 50 rounds, the concepts are rotated among the clients. The three control experiments are

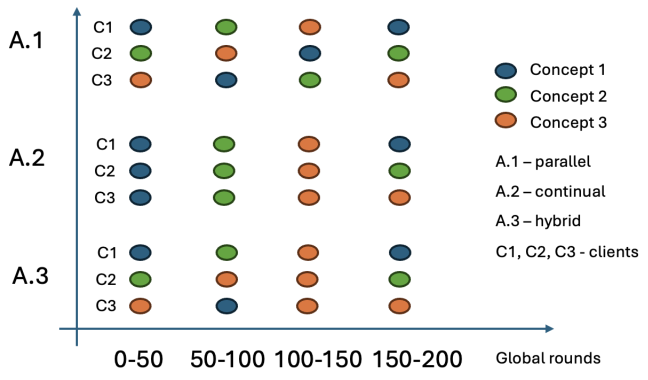


Fig. 2: Illustration of the control experiments conducted. They simulate three different learning and concept drift scenarios: parallel, continual, and hybrid.

described in detail below:

- *Experiment A.*1 simulates parallel learning of the three concepts represented in the clients' data. Concept drift occurs at the client level, where the same concept is distributed to all clients during the same training period. The setup evaluates the model's ability to handle local concept drift, where each client receives different data distributions that periodically change over time. This allows us to analyze the model's ability to adapt to local variations while retaining knowledge from previously learned concepts.
- *Experiment A.*2 simulates continual learning of the three concepts, the drift occurs while the global model is being built. All clients receive the same concept during each interval. Every 50 global rounds, the concept shifts for all clients, leading to concept drift at the global level. In the last 50 global rounds, each client receives a different concept. The goal is to evaluate the model's ability to handle global concept drift, where all clients experience identical data distribution changes at the same time.
- *Experiment A.*3 is a hybrid setup simulating local and global concept drifts. In the first two shifts (rounds 0 to 49 and 50 to 99), clients receive different concepts, experiencing local drift. In rounds 100 to 149, all clients learn the same concept, introducing global drift. A mix of learned local and global concepts is then introduced in the final phase. The goal is to assess the model's ability to simultaneously adapt to both drift types.

Experiments with real-world data are performed on two datasets, 5G and Air Quality, as described below:

- *Experiment* $B.1$ simulates an evolving streaming scenario and is carried out on 5G data. In this scenario, the three clients learn three different concepts, i.e., ElBorn, LesCorts, and PobleSec, each with a different size. These concepts are divided into four equal partitions, see Fig. 1. Each client receives different data every 50 rounds.
- *Experiment* $B.2$ is similar to experiment $B.1$ and simulates a scenario with Air Quality data. Eleven clients each receive a distinct dataset with an equal number of instances representing different concepts. As in $B.1$, the datasets are divided into five equal partitions, and clients train on a new partition every 50 rounds.

In both types of experiments, we studied two different versions of our algorithm, called FedCluLearn and FedCluLearn-Prox, which use two distinct schemes to build the global model. The first version uses the aggregation formula (2) to construct the global model at each training round, while the second employs the optimization proposed in FedProx to address data heterogeneity.

From a convergence perspective, FedCluLearn may struggle in heterogeneous settings due to client drift. FedCluLearn-Prox addresses this issue by incorporating FedProx, which stabilizes training by penalizing deviation from the global model via a proximal term. This has been shown to improve convergence under Non-IID conditions by mitigating local overfitting [17]. Moreover, the proximal term retains the local updates near the global optimization trajectory, which reduces client divergence and accelerates convergence in environments with Non-IID data distributions. Experimental evidence across multiple FL benchmarks supports the effectiveness of this modification, particularly in scenarios with high statistical heterogeneity [17, 14]. These results suggest that the incorporation of FedProx into FedCluLearn leads to more reliable and faster convergence in real-world Non-IID environments.

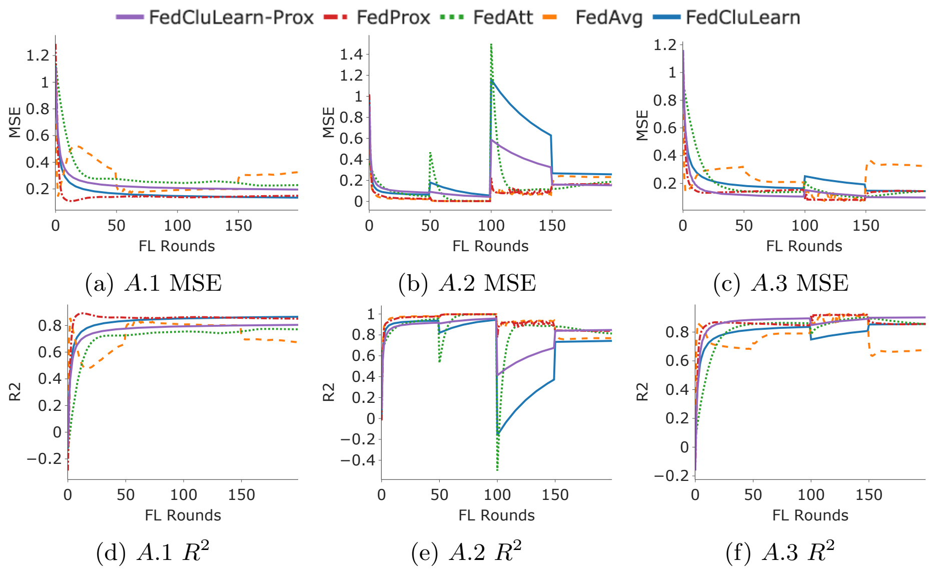
In addition, we conducted an ablation study in which we evaluated different aging schemes, e.g., from using all local modal updates to building the global model only from the updates of the last training round. The former reflects the scenario in which the aging component is omitted, while the latter explores the removal of the component using previously learned concepts. We have used **Mean Squared Error (MSE)** and **R-squared (R^2)** to evaluate the performance of the studied algorithms.

5 Experimental results and discussion

5.1 Control experiments

In the three control experiments ($A.1$, $A.2$, and $A.3$), we examine the two versions of our algorithm mentioned above, FedCluLearn and FedCluLearn-Prox. In addition, three different data aging schemes of the two versions are evaluated in the three experiments.

Fig. 3 shows the performance, evaluated in terms of MSE and R^2, of FedCluLearn and FedCluLearn-Prox, which aggregate all local model updates of the active concepts when building the global model. Their performance is compared to that of the three baseline algorithms. Note that this is an ablation study scenario in which the aging component is omitted. As we can see in *Experiment* $A.1$, FedCluLearn and FedCluLearn-Prox show better robustness to concept drifts that occur at the local level than FedAvg and FedAtt. Furthermore, the performance of FedCluLearn is comparable to that of FedProx, while the latter algorithm outperforms FedCluLearn-Prox. In *Experiment* $A.2$, FedCluLearn and FedCluLearn-Prox are observed to be much more affected by global concept drifts than FedAvg and FedProx, but as can be seen in the last shift, FedCluLearn-Prox remembers the already learned concepts better than the baseline algorithms. Our observations in *Experiment* $A.3$ confirm those of the other two experiments. To understand whether the behavior of FedCluLearn and FedCluLearn-Prox in *Experiment* $A.2$ is due to the SI threshold used (0.5), we also ran this experiment with threshold values of 0.7, 0.8, and 0.9, see Fig. 4. The two versions of our algorithm show different performance patterns with respect to different threshold values. In addition, both algorithms show significantly better performance at a value of 0.9. This is not surprising and is due to



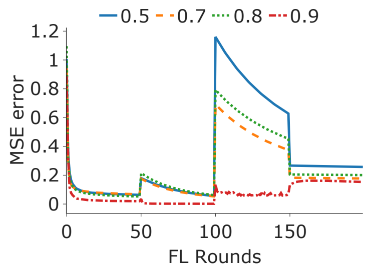
(a) $A.1$ MSE (b) $A.2$ MSE (c) $A.3$ MSE

(d) $A.1$ R^2 (e) $A.2$ R^2 (f) $A.3$ R^2

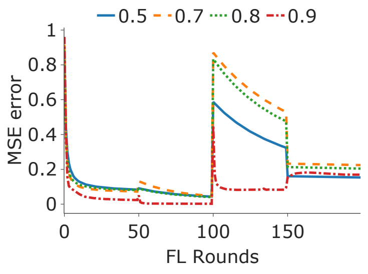
Fig. 3: Results of the evaluation of the global models built by FedAvg, FedAtt, FedProx, FedCluLearn, and FedCluLearn-Prox, in the three experiments $A.1$, $A.2$, and $A.3$. FedCluLearn and FedCluLearn-Prox aggregate all local model updates of the active concepts when building the global model in each training round.

the construction of more clusters, i.e., a high discrimination between different local model updates.

In Fig. 5, we study the FedCluLearn and FedCluLearn-Prox in the three control setups that utilize local model updates only from the last training round of the active concepts when building the global model. This is an ablation study scenario in which the component that uses previously learned concepts has been excluded. In this setting, FedCluLearn-Prox is shown to best handle local and global concept drift and forgetting in all three setups compared to the other evaluated algorithms. FedCluLearn also shows significantly better performance in the case of *Experiment* $A.2$ and the third shift of *Experiment* $A.3$. This is because, in the case of global concept drift, more recent clients' model updates are more beneficial for the global model aggregation to capture the feature of a newly appeared concept. Furthermore, FedCluLearn-Prox outperforms all other algorithms in *Experiment* $A.1$, while FedCluLearn is better than FedAvg and

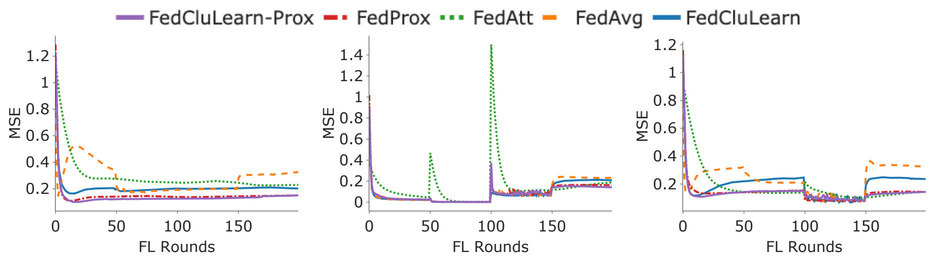


(a) MSE curves of FedCluLearn



(b) MSE curves of FedCluLearn-Prox

Fig. 4: Results of the evaluation of the global models built by FedCluLearn and FedCluLearn-Prox in the experiment $A.2$ with different SI threshold values.



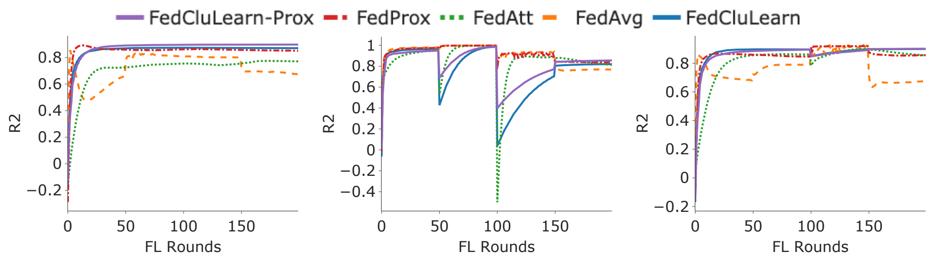
(a) $A.1$ MSE (b) $A.2$ MSE (c) $A.3$ MSE

Fig. 5: Results of the evaluation of the global models built by FedAvg, FedAtt, FedProx, FedCluLearn, and FedCluLearn-Prox, in the three experiments $A.1$, $A.2$, and $A.3$. FedCluLearn and FedCluLearn-Prox use only recent (from the last training round) local model updates of the active concepts when building the global model in each training round.

FedAtt in the same setup. This is most likely due to the fact that the performance of the global models is not negatively affected by the earlier learned immature client updates. Note that the evaluation with respect to R^2 shows similar results and can be seen in our public repository.

Fig. 6 examines the FedCluLearn and FedCluLearn-Prox that aggregate only half of the local model updates of the active concepts when building the global model. Interestingly, this leads to an improved performance of FedCluLearn. Namely, in the current setting, both FedCluLearn and FedCluLearn-Prox outperform the baseline algorithms in *Experiment* $A.1$. In *Experiment* $A.2$, we observe the same performance patterns as in the case where all local model updates are used to build the global model, see Fig. 3. However, the performance of FedCluLearn is significantly improved in *Experiment* $A.3$ compared to the two setups, whose results are shown in Fig. 3 and Fig. 5. We believe this is because only the most recent, mature local modal updates are used to build the global models. The evaluation results with respect to MSE are similar and can be seen in our public repository.

Our main finding from the control experiments is that the performance of both versions of our algorithm, FedCluLearn and FedCluLearn-Prox, is affected by the aging scheme used. In other words, they have shown better performance when the aging scheme included an appropriate number of previously learned concepts. Therefore, both components explored by our ablation study are important for the performance of the algorithms. The effect of the aging scheme on the performance of the algorithms is examined further in our experiments with real-world data. In addition to the above findings, FedCluLearn and FedCluLearn-Prox have shown superior robustness to local concept drift (Experiment $A.1$), while FedCluLearn-Prox has clearly demonstrated better memory retention in global drift scenarios (Experiment $A.2$). The hybrid experiment ($A.3$) has con-



(a) $A.1$ R^2 (b) $A.2$ R^2 (c) $A.3$ R^2

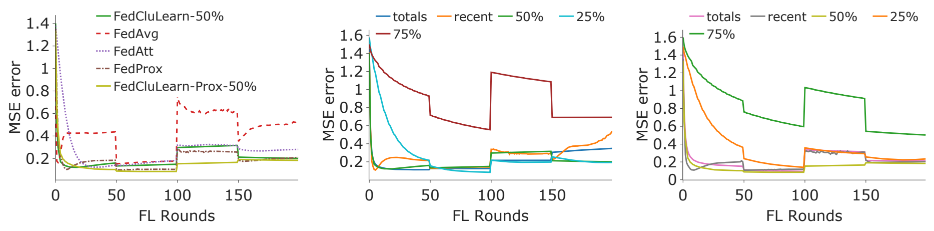
Fig. 6: Results of the evaluation of the global models built by FedAvg, FedAtt, FedProx, FedCluLearn, and FedCluLearn-Prox, in the three experiments $A.1$, $A.2$, and $A.3$. FedCluLearn and FedCluLearn-Prox aggregate only half of the last (most recent) local model updates of the active concepts when building the global model in each training round.

firmed these results, reinforcing the effectiveness of FedCluLearn-Prox in dealing with both types of drift.

5.2 Experiments with real-world data

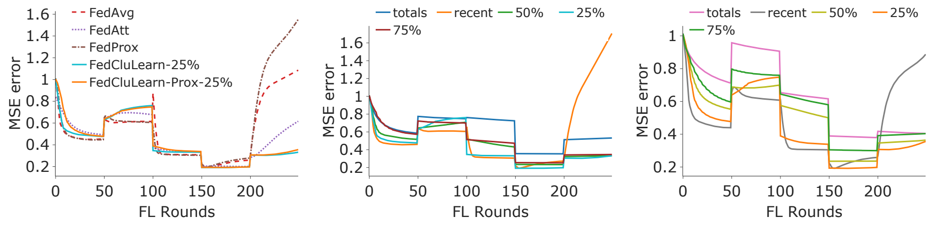
We report and discuss the results of two experiments conducted on real-world data below.

In Fig. 7, we compare the performance of FedCluLearn and FedCluLearn-Prox with that of the three baseline algorithms on 5G data in *Experiment B*.1. We also examine how the performance of our algorithms is affected by the percentage of recent local model updates used to build the global model. In Fig. 7(b), we evaluate the performance of five versions of FedCluLearn, called FedCluLearn-total, FedCluLearn-recent, FedCluLearn-75%, FedCluLearn-50%, and FedCluLearn-25% in terms of MSE. FedCluLearn-total denotes the version that aggregates all local model updates of the active concepts when building the global model in each training round. FedCluLearn-recent, FedCluLearn-75%, FedCluLearn-50%, and FedCluLearn-25% refer to versions that only aggregate local model updates from the last training round and the respective percentage (75%, 50%, and 25%) of local model updates of the active concepts when building the global model. The same comparisons, but for the respective five versions of FedCluLearn-Prox, are shown in Fig. 7(c). Fig. 7(a) then compares the best performing versions of FedCluLearn and FedCluLearn-Prox with the three baseline algorithms. These are FedCluLearn-50% and FedCluLearn-Prox-50%. As we can observe, FedCluLearn-Prox-50% significantly outperforms FedAvg and FedAtt, as well as its relative algorithm, FedProx. FedCluLearn-50% also shows significantly better performance compared to FedAvg, it also outperforms FedAtt and is very close to the performance of FedProx. It is interesting to note that the performance signatures of FedCluLearn-recent (see Fig. 7(b)) and FedAvg (see Fig. 7(a)) have very similar shapes, but the former algorithm shows



(a) MSE-Baselines (b) MSE-FedCluLearn (c) MSE-FedCluLearn-Prox

Fig. 7: Results of experiment B.1 with 5G data simulating an evolving streaming scenario. The middle plot compares the performance of the three versions of FedCluLearn. The same comparison is shown in the right plot, but for FedCluLEarn-Prox. In the left plot, the best performing versions of FedCluLearn and FedCluLearn-Prox are compared to the three baseline algorithms.



(a) MSE-Baselines (b) MSE-FedCluLearn (c) MSE-FedCluLearn-Prox

Fig. 8: Results of experiment $B.2$ with Air Quality data simulating an evolving streaming scenario. The middle plot compares the performance of the five versions of FedCluLearn. The same comparisons are shown in the right plot, but for FedCluLearn-Prox. In the left plot, the best performing versions of FedCluLearn and FedCluLearn-Prox are compared to the three baseline algorithms.

better performance. We believe that this is due to the fact that FedCluLearn uses only local model updates from the currently active concepts to build the global model. A similar trend is observed for FedCluLearn-Prox-recent and FedProx (see Fig. 7(c) and Fig. 7(a)).

In Fig. 8, we compare the performance of FedCluLearn and FedCluLearn-Prox with the three baseline algorithms on the Air Quality data in *Experiment* $B.2$. Similar to *Experiment* $B.1$, we examine the five different versions of each of the two algorithms. In this experiment, the best performing versions of FedCluLearn and FedCluLearn-Prox are FedCluLearn-25% and FedCluLearn-Prox-25%. In Fig. 8(a), FedCluLearn-25% and FedCluLearn-Prox-25% have very similar performance that is better than that of FedAvg and FedAtt and comparable to that of FedProx in most data batches. In addition, both versions of our algorithm handle data drift in the last batch better than the three baselines.

In summary, FedCluLearn has consistently shown better performance than FedAvg and FedAtt, while FedCluLearn-Prox has outperformed the three baselines and FedCluLearn in most experiments. However, both models are sensitive to aging schemes, with suboptimal strategies that compromise global drift adaptation. FedCluLearn-Prox-percentage-50% has shown the best overall performance, significantly outperforming FedAvg, FedAtt, and FedProx. Meanwhile, FedCluLearn-25% and FedCluLearn-Prox-25% have emerged as the best performing variants, confirming that the optimal aging scheme is data dependent.

The five versions of FedCluLearn and FedCluLearn-Prox are also evaluated with respect to R^2 in *Experiments* $B.1$ and $B.2$. The results are similar to those reported under MSE and can be seen in our public repository.

6 Conclusion and future work

In this study, we have proposed a novel Federated Continual Learning (FCL) algorithm, called FedCluLearn, which interprets the federated training process as a stream clustering scenario. It uses the stream micro-cluster indexing scheme to store statistics about local model updates to deal with catastrophic forgetting and concept drift. Additionally, FedCluLearn considers the recency of stored local model updates to determine how many to use in building the global model during each training round.

Two versions of the proposed FCL approach, FedCluLearn and FedCluLearn-Prox, have been evaluated in several control and real-world data experiments, studying various ablation scenarios using different aging schemes and comparing their performance with three baselines, FedAvg, FedProx, and FedAtt. The experimental results have shown that a balanced use of older local model updates can lead to robust handling of concept drift and catastrophic forgetting by both models. FedCluLearn has consistently shown better performance than FedAvg and FedAtt, while FedCluLearn-Prox has outperformed all three baselines, and FedCluLearn in most cases studied.

Future research will focus on dynamic aging schemes that adapt to evolving data distributions to enhance performance in global drift scenarios. To improve algorithm efficiency, we plan to organize cluster statistics on the server using a tree structure like B-trees or R-trees. Additionally, we aim to extend this work to larger, real-world federated environments with more heterogeneous clients to gain insights into FedCluLearn's scalability and applicability. We acknowledge that the current evaluation of the proposed FedCluLearn is primarily based on empirical evidence. Incorporating statistically significant tests is a crucial next step in strengthening the validity of our findings. We also intend to compare FedCluLearn with continual learning methods adapted for FL, in order to provide a more contextualized view of its performance in lifelong learning settings.

References

1. Aggarwal, C.C., Reddy, C.K.: Data Clustering: Algorithms and Applications. Chapman & Hall/CRC, 1st edn. (2013)
2. Aggarwal, C.C., et al.: A framework for clustering evolving data streams. In: Freytag, J.C., et al. (eds.) Proc. VLDB Conf., pp. 81–92. Morgan Kaufmann (2003)
3. Agrahari, S., Singh, A.K.: Concept drift detection in data stream mining: A literature review. Journal of King Saud University-Computer and Information Sciences **34**(10), 9523–9540 (2022)
4. Benzing, F.: Unifying importance based regularisation methods for continual learning. In: Int. Conference on AI and Statistics. pp. 2372–2396. PMLR (2022)
5. De Lange, M., et al.: A continual learning survey: Defying forgetting in classification tasks. IEEE trans. on patt. anal. and mach. intell. **44**(7), 3366–3385 (2021)
6. Estiri, A.H., Maheswaran, M.: Attentive federated learning for concept drift in distributed 5g edge networks. ArXiv **abs/2111.07457** (2021)
7. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
8. Gama, J.a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (2014)
9. Gepperth, A., Hammer, B.: Incremental learning algorithms and applications. In: European symposium on artificial neural networks (ESANN) (2016)

10. Ghosh, A., et al.: An efficient framework for clustered federated learning. Advances in neural information processing systems **33**, 19586–19597 (2020)
11. Islam, M.S., et al.: Fedclust: Tackling data heterogeneity in federated learning through weight-driven client clustering. In: Proc. of ICPP'24. pp. 474–483 (2024)
12. Iwashita, A.S., Papa, J.P.: An overview on concept drift learning. IEEE access **7**, 1532–1547 (2018)
13. Jiang, X., Borcea, C.: Concept matching: clustering-based federated continual learning. arXiv preprint arXiv:**2311.06921** (2023)
14. Karimireddy, S., Kale, et al. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. *Proceedings Of The 37th International Conference On Machine Learning.* **119** pp. 5132-5143 (2020,7,13)
15. Kranen, P., et al.: The clustree: indexing micro-clusters for anytime stream mining. Knowledge and Information Systems **29**, 249–272 (2011)
16. Lesort, T., et al.: Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. Information fusion **58**, 52–68 (2020)
17. Li, T., et al.: Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems **2**, 429–450 (2020)
18. Lin, G., Chu, H., Lai, H.: Towards better plasticity-stability trade-off in incremental learning: A simple linear connector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 89–98 (2022)
19. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. vol. 5, pp. 281–298 (1967)
20. McMahan, B., et al.: Communication-efficient learning of deep networks from decentralized data. In: Proc. of the 20th AISTATS (2017)
21. Ouyang, X., et al.: Clusterfl: A clustering-based federated learning system for human activity recognition. ACM Trans. on Sensor Networks **19**(1), 1–32 (2022)
22. Perifanis, V., et al.: Towards energy-aware federated traffic prediction for cellular networks. In: 2023 8th Int. Conf. on FMEC. pp. 93–100 (2023)
23. Pfeiffer, J., et al.: Modular deep learning. ArXiv **abs/2302.11529** (2023)
24. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Comp. and Applied Mathematics **20**, 53–65 (1987)
25. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. Advances in neural information processing systems **30** (2017)
26. Wadewale, K., Desai, S.: Survey on method of drift detection and classification for time varying data set (2015)
27. Wang, L., et al.: Memory replay with data compression for continual learning. arXiv preprint arXiv:**2202.06592** (2022)
28. Wang, L., et al.: A comprehensive survey of continual learning: Theory, method and application. IEEE Trans. on Pattern Analysis and Machine Intelligence (2024)
29. Wang, Z., et al.: Federated continual learning for edge-ai: A comprehensive survey. arXiv preprint arXiv:**2411.13740** (2024)
30. Yang, X., et al.: Federated continual learning via knowledge fusion: A survey. IEEE Transactions on Knowledge and Data Engineering **36**(8), 3832–3850 (2024)
31. Yoon, J., Madaan, D., Yang, E., Hwang, S.J.: Online coreset selection for rehearsal-based continual learning. arXiv preprint arXiv:**2106.01085** (2021)
32. Zhang, S., et al.: Cautionary tales on air-quality improvement in beijing. Proc. of the Royal Society A: Mathematical, Physical and Engineering Sciences **473** (2017)