Asynchronous Federated Continual Learning

Donald Shenaj, Marco Toldo, Alberto Rigon, Pietro Zanuttigh
University of Padova, Italy

Abstract

The standard class-incremental continual learning setting assumes a set of tasks seen one after the other in a fixed and pre-defined order. This is not very realistic in federated learning environments where each client works independently in an asynchronous manner getting data for the different tasks in time-frames and orders totally uncorrelated with the other ones. We introduce a novel federated learning setting (AFCL) where the continual learning of multiple tasks happens at each client with different orderings and in asynchronous time slots. We tackle this novel task using prototype-based learning, a representation loss, fractal pre-training, and a modified aggregation policy. Our approach, called FedSpace, effectively tackles this task as shown by the results on the CIFAR-100 dataset using 3 different federated splits with 50, 100, and 500 clients, respectively. The code and federated splits are available at https://github.com/LTTM/FedSpace.

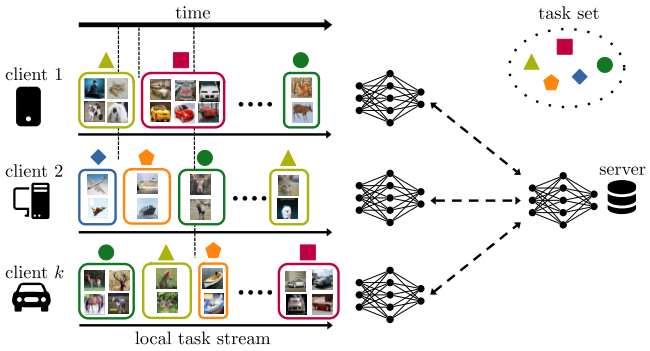


Figure 1. Illustration of the considered asynchronous federated continual learning setting (AFCL). Each client follows his own task stream, experiencing the global set of available tasks in different time slots. At the end of each round, the server aggregates clients trained on heterogeneous sets of classes.

1. Introduction

Federated Learning (FL) involves distributed learning across multiple heterogeneous devices, coordinated by a central server, without sharing private data. In its standard fashion, FL implies a static configuration, where clients have access to the same local data and do not change their tasks for the entire training process. Furthermore, the training procedure is typically divided into rounds perfectly aligned across the different clients.

This setting, even if reasonable for research purposes, does not represent a realistic real-world learning setting where the clients (e.g., smartphones, autonomous cars, etc..) typically acquire data from onboard cameras or other devices while the training goes on and can be required to learn new tasks during time. Furthermore, each client asynchronously works independently from the others and can get new data at any time typically misaligned with the others.

Recently the continual learning (CL) paradigm has emerged as a more practical setting to mimic a real-world learning process, where new data and tasks are likely to be encountered over time. We, therefore introduce a new setting combining continual and federated learning. We allow each client to follow its own continual learning path in an asynchronous way with respect to the other clients, while the server orchestrates the distributed training in order to safely aggregate the different learned models coming from the streams of information locally acquired at the clients.

More in detail, we propose to tackle a comprehensive setting, that we called Asynchronous Federated Continual Learning (AFCL), in which each client of a federated system undergoes its own continual learning procedure, progressing task by task in an independent fashion, *i.e.*, by:

- following its individual sequence of tasks, which can be different from client to client,
- under its own temporal scheduling, i.e., the global training comprises out-of-sync local streams of data corresponding to different tasks.

The overall objective of AFCL is to end up with a prediction model that has assimilated all the tasks' knowledge acquired by individual clients, avoiding catastrophic forgetting inherent in the incremental local task progression.

We identify the main challenges of the AFCL setting in:

- **Asynchronous Learning**: Clients see different tasks at the same time, and switch between tasks in a misaligned manner.
- **Global CL**: Even though globally multiple tasks are experienced simultaneously, there will be tasks not currently observed, so it is crucial to preserve the information related to them.

To address this task we resort to a Federated learning System with Prototype Aggregation for Continual rEpresentation (abbreviated as FedSpace), which exploits class prototypes in feature space and contrastive learning both to preserve previous knowledge and to avoid too divergent behavior between the different clients.

The main contributions of this paper are the following:

- We address AFCL, a novel federated learning setting where clients explore different tasks in time-frames and orders totally uncorrelated with the other clients.
- We propose 3 realistic federated splits for the CIFAR-100 dataset, using 50, 100, and 500 clients.
- We evaluate the effectiveness of pre-training in federated learning using fractal images.
- We introduce an aggregation strategy for the prototypes and resort to prototype augmentation loss to tackle catastrophic forgetting, as done in continual learning [21].
- We design a contrastive representation loss able to align the old aggregated prototypes coming from other clients, with their new locally learned representation while disentangling it from different classes.
- We adopt a modified model aggregation strategy that combines the client models with the previous server model to keep under control the model drift in this very challenging setting.

2. Related Works

Federated Learning. Federated Learning (FL) is a machine learning paradigm introduced in [20] as an alternative way to train a global model from a federation of devices keeping their data local, and communicating to the server only the model parameters. The iterative FedAvg algorithm [20] represents the standard approach to address FL. First, the server randomly selects a subset of clients, which will perform training on their local data and send back to the server their updated models. Then, the server computes the global model as the weighted mean of the received models and repeats the previous step for many rounds. Despite its simplicity, FedAvg empirically demonstrates remarkable results. In the context of systems heterogeneity, FedAvg does not allow participating devices to perform variable amounts of local work based on the constraints of their underlying system.

This issue is addressed by FedProx [18], which can be seen as a generalization and re-parametrization of FedAvg, providing a theoretical convergence guarantee when learning over data from non-identical distributions.

SCAFFOLD [14] can be seen as an improved version of the distributed optimization algorithm [25] where a fixed number of (stochastic) gradient steps are used in place of a proximal point update. It deals with the problem of client drift by introducing control variates. In particular, the difference between the local control variate and the global control variate is used to correct the gradients in local training.

However, both FedProx and SCAFFOLD, do not provide advantages over FedAvg when training deep learning models on image datasets. This motivates MOON [17] to propose a new approach for handling non-IID image datasets. It proposes to enforce similarity between model representations to correct the local training of individual parties, conducting contrastive learning at model-level.

Some works [32, 33] consider an asynchronous setting where each heterogeneous client sends the model update as soon as the training is finished and the aggregation at the server side is done using information from a single client at a time. This setup differs from our setting since in our case the asynchronously is not in the time slot but in the task progression and update while we perform the aggregation with multiple clients at a time as in the standard setting.

Continual Learning. Deep neural networks experience catastrophic forgetting when trained on new data [9]. The objective of continual learning, therefore, is to enable a model to learn from a never-ending stream of data without the need to retrain the model from scratch every time new data become available.

For practical computer vision applications, it has been extensively studied in a class incremental fashion [5, 23]. A first direction to mitigate catastrophic forgetting is to store and replay samples of old classes [2, 8, 24, 31]. Another direction is to build a separate generative model able to reproduce old samples, and reuse them while training the model on new classes [19, 27, 30].

Prototype-based learning has also been widely used for continual learning [21, 28, 37]. In PASS [37] a simple non-exemplar based technique is proposed to address the catastrophic forgetting problem in incremental learning. Instead of memorizing raw images, they memorize one class-representative prototype for each old class and adopt prototype augmentation by adding Gaussian noise in the feature space to maintain the decision boundary of previous tasks.

Federated Continual Learning. Very recently, some works started to address federated learning in settings, where the learned tasks change over time [7, 13, 35, 36], and where the clients' models are adapted to different domains [26, 34]. Focusing on settings where the task changes over time, i.e., on continual learning, some recent interesting works can be pointed out.

A preliminary work [35] exploits the storage of training samples at the server side, compromising the privacy of the clients' data. In FedWEIT [36] the network weights are decomposed into global federated parameters and sparse task-specific parameters. Each client receives selective knowledge from other clients by taking a weighted combination of their task-specific parameters. However, they still require a buffer for data replay and does not scale when increasing the number of clients or tasks, due to the communication overhead. The task of Federated Class-Incremental Learning is addressed in [6, 7] with a focus on the privacy of the systems: differently from previous works, they only store perturbed images in a replay memory, while the server has no prior knowledge about when local clients will receive the data for new classes.

Another federated continual learning scenario is considered in [12]. Here, the server is pre-trained on a set of classes, corresponding to the first task. Then, each client learns the remaining ones by following its own task stream. Differently from the proposed AFCL setting, clients explore new sets of classes synchronously.

FCCL [13] deals with catastrophic forgetting and heterogeneity problems in federated learning, by leveraging unlabeled public data. In their setting, "continual" refers to the fact that clients continually learn from each other as the training proceeds.

3. Problem Formulation

Let us consider a federated learning setting where N clients tackle a classification problem aiming at assigning each data sample (*i.e.*, an image in the considered setting) to one of the possible classes $c \in C$. We assume to use a generic deep learning model $\mathcal{M} = D \circ E$, composed of an encoder E followed by a decoder D as most architectures for image classification.

In the considered setting each client at each time step has access to a different subset of the data: let $C_k^{(t)} \subset C$ denote the set of classes (*i.e.*, the task) available for client k at round t. We denote with $\theta^{(t)} = \{\xi^{(t)}, \psi^{(t)}\}$ the parameters of the encoder and decoder models at round t. We assume that the client has access to the same data stream for a certain time interval of variable length, then moves to a new set of data, and so on. We can thus define an ordered set of boundaries $T_k = \{T_{k,i}\}_{i=1}^{R_k}$, such that each client changes task only at the boundaries location, i.e., $C_k^{(t)} = C_k^{(T_{k,i})} \quad \forall t : T_{k,i-1} < t \leq T_{k,i}$. R_k denotes the total number of data streams for client k (see Fig. 2).

Algorithm 1: FedSpace learning scheme

Input: Number of communication rounds T,
N clients each with its data stream
$\mathcal{D} = \{\mathcal{D}_k^{(T_{k,i})}\}_{i=1}^{R_k}$
Output: Model weights $\theta^{(T)}$
Pre-training
  Build a fractal dataset D_f
  Train the initial server model $\theta^{(0)}$ on $\mathcal{D}_f$
Federated training
  $e_{k,c}^{(t)}, r_k^{(t)} = \emptyset, \emptyset$
  for *round* $t \in T$ **do**
    for *client* $k \in K$ *in parallel* **do**
      Set $\theta_k^{(t)} = \theta^{(t-1)}$
      $e_{k,c}^{(t)}, r_k^{(t)} \leftarrow$ ComputeProto($\mathcal{D}_k^{(t)}$) (Eq. 4, Eq. 5)
      Optimize $\theta_k^{(t)}$ with:
      $\mathcal{L} = \mathcal{L}_{CE} + \lambda_p \mathcal{L}_p + \lambda_r \mathcal{L}_r$ (Eq. 3)
    $\theta^{(t)}$ = AggrModel($\theta_k^{(t)} \quad \forall k \in K, \theta^{(t-1)}$) (Eq. 10)
    $e_c^{(t)}, r^{(t)} \leftarrow$ AggrProto($e_{k,c}^{(t)}, r_k^{(t)} \forall k \in K$) (Eq. 6)

Each client has a local stream dataset $\mathcal{D} = \{\mathcal{D}_k^{(T_{k,i})}\}_{i=1}^{T}$, where $\mathcal{D}_k^t = \{\mathbf{X}, \mathbf{Y}\}$ corresponds to the image-label pair. Note how the available data change at each boundary $T_{k,i}$ as depicted in Fig. 2. Each client performs local training on its sequence of tasks $\{C_k^{(T_{k,i})}\}_{i=1}^{R_k}$ in an asynchronous way w.r.t the others (see Fig. 1 and Fig. 2).

At each communication round t, a subset of clients $K_t \subset N$ is randomly selected. Every client $k \in K_t$ downloads the global model (i.e., the model weights $\theta^{(t-1)}$), performs training on his local data $\mathcal{D}_k^{(t)}$, and sends the updated weights $\theta_k^{(t)}$ to the server, which will perform aggregation and produce the new global model $\theta^{(t)}$.

4. Method

In this section, we are going to describe in detail the proposed method (FedSpace) to tackle AFCL. Firstly the pre-training step at the server side exploiting fractal images is discussed in Sec. 4.1. Then, the optimization framework for each client is introduced in Sec. 4.2. The prototype aggregation and representation loss are detailed respectively in Sec. 4.3 and Sec. 4.4. Finally, the server aggregation is discussed in Sec. 4.5.

The proposed framework is depicted in Fig. 3 and summarized in Algorithm 1.

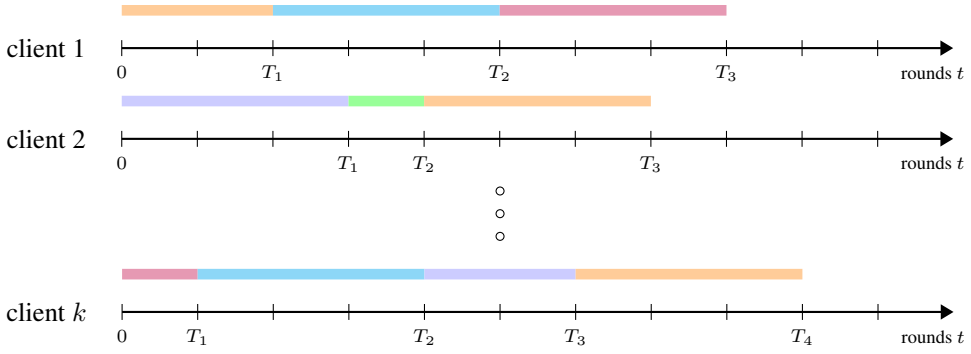


Figure 2. Example timeline showing the progression of the different tasks on the various clients. (*Best viewed in colors)*

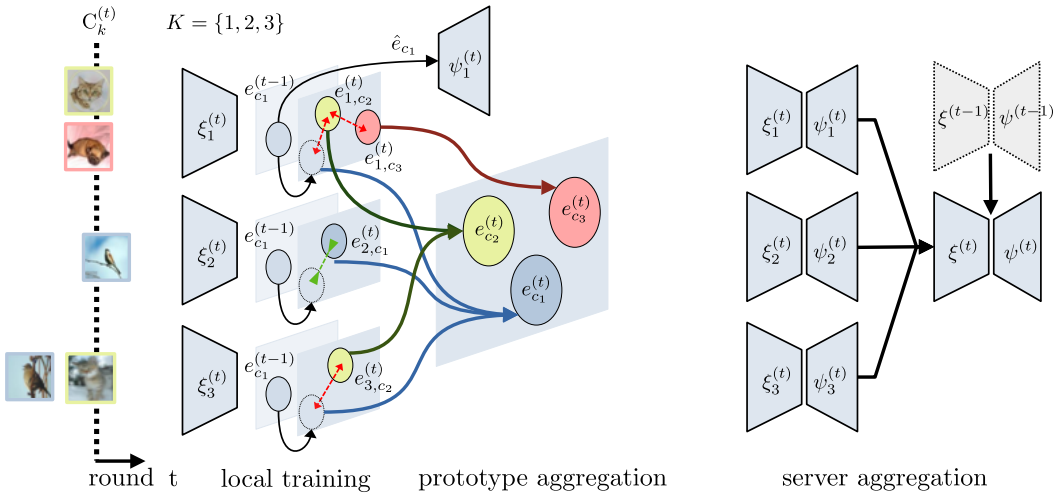


Figure 3. Overview of the FedSpace learning algorithm, proposed for solving AFCL.

4.1. Server Fractal Pre-training

Recent studies suggest that pre-training allows to improve performances for the downstream task and speeds up training in federated learning [4, 22]. We resort to the pre-training strategy with fractal images proposed in [1] for a centralized setting, and use it to pre-train the server model before starting the learning at clients' side. Fractal images are generated from affine Iterated Function Systems (affine IFS) [15], therefore they provide perfect label accuracy at zero cost, no privacy issues, and most importantly do not contain images or classes present in the CIFAR-100 dataset, hence they are the best candidate in FL. Additionally, in AFCL, since clients experience new classes inside different time slots, pretraining allows to limit client drift and achieve a better alignment of their internal representations since they all start from an initial common feature space. Following [1], we consider a set of classes C_f (with $|C_f| = 1000$, that is higher than the number of classes $|C|$ used in our experiments) and build a dataset $\mathcal{D}_f = \{X_f, Y_f\}$ where $x_f \in X_f$ is a fractal image generated from the $y_f \in Y_f$ IFS code or label. Hence, training is performed using the standard cross-entropy loss for multi-class classification. At the end of the pre-training, the server parameters $\theta^{(0)} = \{\xi^{(0)}, \psi^{(0)}\}$ are sent to every client $k \in K$, which will proceed to update their model. Notice that the number of local classes at client k, $|C_k^{(0)}|$ is smaller than the number of classes $|C_f|$ used during pre-training. Hence, let $\psi_{nc}^{(0)}$ be the classifier weights $\forall c \notin C_k^{(0)} \cap C_f$, we can divide the decoder parameters into the ones corresponding to the not currently considered classes $\psi_{nc}^{(0)}$ and all the others $\hat{\psi}^{(0)}$:

$$\psi^{(0)} = \{\hat{\psi}^{(0)}, \psi_{nc}^{(0)}\}, \tag{1}$$

and initialize each client model θ_k^0 as follows:

$$\{\xi_k^{(0)}, \psi_k^{(0)}\} \leftarrow \{\xi^{(0)}, \hat{\psi}^{(0)}\}. \tag{2}$$

4.2. Clients' Optimization

Each client performs local training optimizing the following loss function:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_p \mathcal{L}_p + \lambda_r \mathcal{L}_r \tag{3}$$

where $\mathcal{L}_{CE}$ is the standard supervised cross-entropy loss used for classification on $\mathcal{D}_k^{(t)}$, $\mathcal{L}_p$ is the prototype-based loss and finally $\mathcal{L}_r$ corresponds to our contrastive representation loss. The λ_p and λ_r hyper-parameters are used as weighting factors for the respective losses.

4.3. Prototype Aggregation

At each round t, every selected client $k \in K$, for each class $c \in C_k^{(t)}$, computes the mean feature vector, i.e., prototype of the class:

$$e_{k,c}^{(t)} = \frac{1}{|\mathcal{D}_k^{(t)}|} \sum_{\forall c \in C_k^{(t)}} E(\mathbf{X_k^{(t)}}; \xi_{\mathbf{k}}^{(\mathbf{t})}), \tag{4}$$

where $E(\cdot)$ is output of the encoder with weights $\xi_k^{(t)}$ given $\mathbf{X_k^{(t)}}$ in input. Each client has also a radius $r_k^{(t)}$ capturing the variance of the associated features [37]:

$$r_k^{(t)} = \sqrt{\frac{1}{|\mathcal{D}_k^{(t)}|} \sum_{\forall c \in C_k^{(t)}} \frac{Tr(\Sigma_c^{(t)})}{d}}, \tag{5}$$

where $Tr(\cdot)$ denotes the trace operator of a matrix, $\Sigma_c^{(t)}$ is the covariance matrix for the features of class c at round t for client k, and d is the dimension of the feature space.

Then, each client sends the prototype-radius pair $(e_{k,c}^{(t)}, r_k^{(t)})$ to the server which will proceed to aggregate them into global prototypes $(e_c^{(t)}, r^{(t)})$ by computing their weighted average. When at the server side an older representation $e_c^{(t-1)}$ already existed for a specific class c (*i.e.*, at least one client m selected at round $t_0 < t$ performed training with $c \in C_k^{(t_0)}$), it is updated with a moving average with weight β. Therefore, the prototype aggregation becomes:

$$e_c^{(t)} = \begin{cases} \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\mathcal{D}} e_{k,c}^{(t)} & \text{if } c \notin C^{(t-1)} \\ \beta \left(\sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\mathcal{D}} e_{k,c}^{(t)} \right) + (1-\beta) e_c^{(t-1)} & \text{otherwise}. \end{cases} \tag{6}$$

During local training, each client k performs prototype augmentation [37] on the global prototypes $e_c^{(t-1)}$ of classes $c \notin \mathcal{C}_k^{(t)}$ not present in the current learning stage by adding Gaussian noise scaled by the radius $r^{(t-1)}$:

$$\hat{e}_c = e_c^{(t-1)} + r^{(t-1)} * \mathcal{N}(0,1) \tag{7}$$

Finally, a vector of augmented prototypes $\hat{\mathbf{e}}$ of the same length of the batch size is constructed (for each element $\hat{\mathbf{e}}[n]$ its class $c = \mathcal{Y}[n]$ is randomly selected among the ones not present in the current stage). The loss $\mathcal{L}_p$ is thus computed as the cross entropy between the prediction computed from the augmented prototypes and their labels:

$$\mathcal{L}_p = \sum_n \mathcal{L}_{CE}(D(\hat{\mathbf{e}}[n]; \psi_k^{(t)}), \mathcal{Y}[n]). \tag{8}$$

4.4. Representation Loss

Previous works showed that contrastive learning can be efficiently used in continual learning to tackle catastrophic forgetting [3, 21, 29]. Thus, we introduce an additional loss function based on a contrastive paradigm that encourages feature vectors of the same class to be close together while pushing away from each other feature vectors belonging to different classes. At round t, each non-empty global prototype $e_i^{(t-1)}$ corresponding to a generic class $i \in C^{(t-1)} \setminus \mathcal{C}_k^{(t)}$ is augmented according to Eq. (7). Note that $C^{(t-1)} \setminus \mathcal{C}_k^{(t)}$, $C^{(t-1)} \subset C$, represents the set of classes globally discovered by the server up to round $t-1$ but excluding the ones currently being learned. For these classes, a prototype has already been computed and an augmented prototype vector can be computed from Eq. (7). Finally, the augmented prototype vectors and the feature vectors computed on $\mathcal{C}_k^{(t)}$ are concatenated to form a new set of feature vectors.

For every class $c \in \mathcal{C}_k^{(t)}$, a positive and a negative set of feature vectors are constructed. The positive set is represented by a matrix of pairwise cosine similarities between the feature vectors belonging to the considered class, both computed from input data or from augmented prototypes. The negative set is instead computed using feature vectors of all the other classes (again both from input or augmented prototypes).

Formally, the representation loss $\mathcal{L}_r$ is defined as:

$$\mathcal{L}_r = -\frac{1}{|\mathcal{D}_k^{(t)}|} \sum_{c=1}^{|\mathcal{C}_k^{(t)}|} \frac{1}{N_c(N_c-1)} \sum_{i \neq j} \log \frac{e^{s_{i,j}^+}}{e^{s_{i,j}^+} + \sum_{k=1, k \neq c}^{C} e^{s_{i,k}^-}}, \tag{9}$$

where N_c is the number of samples of the class c, $s_{i,j}^+$ is the cosine similarity between the i-th and j-th feature vectors in the positive set, $s_{i,k}^-$ is the cosine similarity between the i-th feature vector in the positive set and the k-th vector in the negative set. The loss is computed as the negative

log-likelihood of the positive set relative to the negative set averaged over the set of classes $\mathcal{C}_k^{(t)}$.

The loss is minimized by adjusting the feature vectors to minimize the distances inside the positive set and maximize the distances between the positive set and the negative set elements.

In this way, the loss enforces each client to learn locally new classes, clustering their features in a consistent way with their past representation obtained from the server (and therefore from the other clients), while at the same time enforcing separation between features of different classes. This allows to obtain a feature clustering that is consistent across different clients and across time slots.

4.5. Server Aggregation

At the end of each communication round, each client sends the model to the server, where the aggregation is performed. Typically in standard FedAvg [20] the server model is updated with the weighted average of the model parameters of each client participating in that round, by weighting his contribution depending on the number of images seen in that round. In our setting, there is no guarantee that all the classes are present in the considered step and furthermore some clients could have just started the learning with the new data stream and have a not very reliable model. In order to stabilize the aggregation avoiding drifts due to unreliable clients and the forgetting of old classes, we perform a weighted average between the newly computed model and the previous model present at the server side. Mathematically:

$$\theta^{(t)} = \rho \left(\sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\sum_{i=1}^{k} |\mathcal{D}_i|} \theta_k^{(t)} \right) + (1-\rho)\theta^{(t-1)}, \quad (10)$$

where θ denotes the server model weights, θ_k denotes the model weights of the client k, $|\mathcal{D}_k|$ is the number of images of client k, and t corresponds to the communication round. We experimentally found that setting $\rho = 0.5$, i.e., a simple average, provides the best performances.

5. Experiments

5.1. Experimental Setup

Datasets. To evaluate the performance of our method, we run the simulations on the CIFAR-100 dataset. [16]. We introduce 3 strongly non-IID splits with 10 tasks each made of 10 classes, while the total number of clients is equal to 50, 100, and 500, respectively. To create them, first, a distribution of the number of samples for each client is computed using a power-law distribution. Next, for each client, the partitioning is done based on a Dirichlet distribution, where the proportions of samples from each class are randomly sampled from the distribution. The parameter of the Dirichlet distribution α was set equal to 3. Each client sets his task streams and boundaries (*i.e.*, number of rounds per task) randomly and independently from other clients.

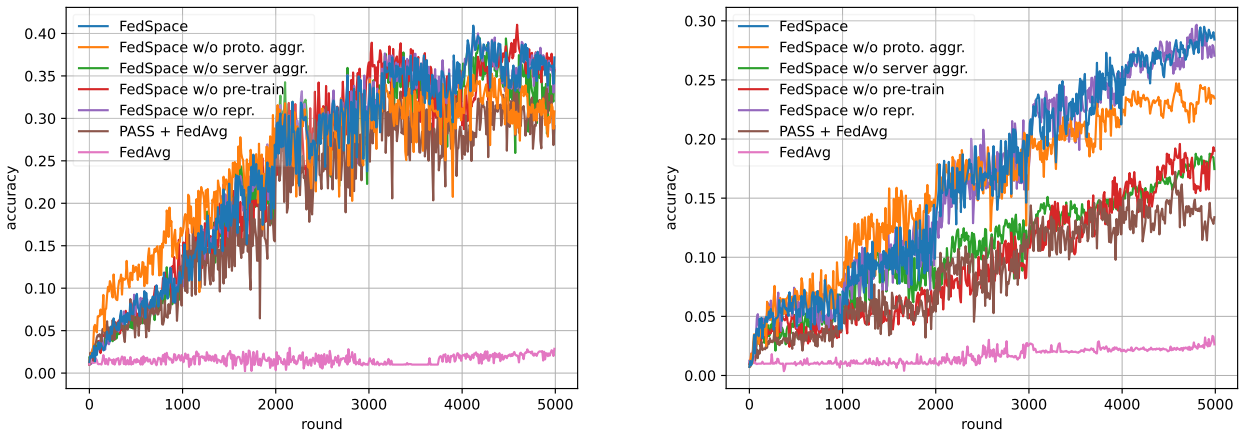
Competitors. Given the novelty of the setting, there is no other work explicitly developed for the task. However, we compared our proposed method with a baseline, e.g., FedAvg [20], that is the standard approach for FL and with PASS [37], a state-of-the-art non-exemplar algorithm for class incremental learning, purposefully adapted to the considered FL setting. In practice, it computes local prototypes at each round and makes use of a prototype-based loss. The aggregation at the server side is done using FedAvg.

Implementation details. The code is implemented in PyTorch, starting from the FedML [10] framework. The employed classification network is ResNet-18 [11]. We run the experiments for 5000 communication rounds, $|K| = 5$ clients per round. The employed optimizer is ADAM, we used a batch size equal to 64, and a learning rate $l_r = 1e{-}3$, which is divided by 2 every 1000 rounds. Concerning the parameters of our method we used $\beta = 0.1$, $\lambda_p = 1e-2$, $\lambda_r = 1e-2$. We also employed the self-supervised label augmentation of [37]. We run the pre-training for one epoch with a batch size of 32, with images of size 256×256. Training is performed sequentially on each client, and run on a single NVIDIA RTX 3090. The code, federated splits, and configuration files are publicly available at https://github.com/LTTM/FedSpace.

5.2. Results

In Table 1, we report the Top-1 accuracy, for each federated split (50, 100, and 500 clients), and compare the performance with FedAvg [20] and PASS [37] adapted to FL. We notice that in this highly challenging scenario, FedAvg [20] is not a feasible solution and provides very poor performances between 2% and 3% (even by adding the fractal pre-training stage of Section 4.1 results do not improve). The main reason is the catastrophic forgetting, here experienced both in a continual learning and in a federated learning way. At each communication round, due to client drift and misaligned learning of new classes across the selected clients, the global model loses knowledge of the previously learned classes. Moreover, when a set of classes is no more experienced by clients selected in future rounds, those are forgotten as well known in continual learning.

Since a continual learning scheme is fundamental to tackle this setting, we considered a state-of-the-art approach for this task, i.e, PASS [37] and combined it with the FedAvg aggregation scheme. We select this approach since it has impressive performances, does not use exemplar memory, and is based on prototypes as our approach. Compared to FedAvg the performance improvement is large. With 50 clients it achieves 29.1%, a reasonable score in this very challenging setting. When increasing the num-



(a) 50 clients.

(b) 500 clients.

Figure 4. Classification accuracy computed at every communication round. Refer to Table 2 for the notation.

Method	Top-1 Accuracy (%)		
	50	100	500
FedAvg [20]	2.86	2.01	2.63
PASS [37] + FedAvg [20]	29.08	22.97	13.39
Ours (FedSpace)	**37.18**	**29.99**	**28.42**

Table 1. Comparison of different methods for 50, 100, and 500 clients splits of CIFAR-100. Top-1 accuracy.

ber of clients, however, the performance drop is significant, achieving around 23% with 100 clients and 13.4% with 500. Notice how especially in the latter case the drop is very relevant, showing that the approach has issues in scaling to a large number of clients.

Our approach achieves 37.2% with 50 clients, outperforming PASS of more than 8%. Furthermore with 100 and 500 clients it achieves 30% and 28.4%, respectively. Especially the latter case shows how our approach scales much better with the number of clients more than doubling the performance of PASS in the most challenging setting.

5.3. Ablation Studies

In order to validate the performance and robustness of the proposed method, we perform an extensive ablation on all the three settings. In Table 2, we report the Top-1 accuracy, computed on all classes C after 5000 rounds, while in Fig. 4 we show the classification accuracy while the training goes on, i.e., at the end of every communication round, for 50 and 500 clients. The Figure reports the accuracy during training for the competitors, our method, and its reduced versions obtained by removing each component singularly.

Interestingly, when the number of total clients is low, during the first rounds keeping the prototypes local (i.e., not aggregating them) leads to higher performances, however, after a certain amount of rounds, aggregating prototypes starts to be beneficial and gives a similar improvement in every setting, with an increase of 6%, 4% and 5% for 50, 100 and 500 clients, respectively (second row of Table 2). Even if it is less evident, this is true also for the representation loss, with an increase of 1%, 3% and 1% (fourth row of Table 2).

When removing the modified server aggregation (*i.e.*, performing aggregation as in FedAvg), the performance decrease by 1.5% for 50 clients, up to 11% with 500 clients. Similarly, by removing the pre-training step with 50 clients, the drop is 1.5%, while when using 500 clients it determines a larger decrease of 10%.

Proto Aggr.	Pre-train	Repr. Loss	Server Aggr.	Accuracy (%) 50	100	500
✓				28.76	29.58	23.45
	✓	✓	✓	30.82	25.89	23.45
✓		✓	✓	35.72	25.89	19.01
✓	✓		✓	35.96	26.84	27.05
✓	✓	✓		35.60	31.90	17.46
✓	✓	✓	✓	37.18	29.99	28.42

Table 2. Ablation of our method on CIFAR-100 for 50, 100 and 500 clients. Top-1 accuracy. Proto aggr. refers to the prototype aggregation (Eq. 6), Pre-train is the server fractal pre-training (Sec 4.1), Repr. Loss is the representation loss (Eq. 9) and Server Aggr. corresponds to the modified aggregation (Eq. 10) instead of FedAvg.

6. Conclusions

In this paper, we introduced a novel realistic setting for asynchronous federated continual learning (AFCL) detailing the main challenges. Then, we presented our framework (FedSpace) which starts by performing pre-training at the server side with fractal images. Next, local training on clients is driven by a loss applied to the augmented old global prototypes and by a contrastive representation loss used to properly allocate the prototypes of new (global) classes while at the same time aligning the new local classes with their past location. Finally, a modified server aggregation strategy reduces client drift. To evaluate the performance of FedSpace, we proposed 3 federated splits of the CIFAR-100 dataset, using 50, 100, and 500 clients, that we used for the experimental evaluation. Our approach reached state-of-the-art results when compared with competing methods adapted to our setting.

Further research will be devoted to the development of more advanced aggregation strategies at the server side and to the combination of the proposed approach with other state-of-the-art continual learning methods.

Acknowledgment

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

References

[1] Connor Anderson and Ryan Farrell. Improving fractal pre-training. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2022. 4

[2] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018. 2

[3] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525, 2021. 5

[4] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. In *The Eleventh International Conference on Learning Representations*, 2023. 4

[5] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 2

[6] Jiahua Dong, Yang Cong, Gan Sun, Yulun Zhang, Bernt Schiele, and Dengxin Dai. No one left behind: Real-world federated class-incremental learning. *arXiv preprint arXiv:2302.00903*, 2023. 3

[7] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10164–10173, 2022. 3

[8] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020. 2

[9] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 2

[10] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020. 6

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[12] Sean M. Hendryx, KC DharmaRaj, Bradley L. Walls, and Clayton T. Morrison. Federated reconnaissance: Efficient, distributed, class-incremental learning. *ArXiv*, abs/2109.00150, 2021. 3

[13] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153, 2022. 3

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020. 2

[15] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 4

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[17] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021. 2

[18] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020. 2

[19] Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *Proceedings of the IEEE/CVF*

International Conference on Computer Vision (ICCV), pages 7026–7035, October 2021. 2

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017. 2, 6, 7

[21] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1114–1124, June 2021. 2, 5

[22] John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? exploring the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2206.15387*, 2022. 4

[23] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019. 2

[24] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 2

[25] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014. 2

[26] Donald Shenaj, Eros Fanì, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 444–454, January 2023. 3

[27] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017. 2

[28] Marco Toldo and Mete Ozay. Bring evanescent representations to life in lifelong class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16732–16741, 2022. 2

[29] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023. 5

[30] Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[31] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 2

[32] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019. 2

[33] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*, 2021. 2

[34] Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. Federated multi-target domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1424–1433, 2022. 3

[35] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1736–1740. IEEE, 2020. 3

[36] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR, 2021. 3

[37] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021. 2, 5, 6, 7