A distillation-based approach integrating continual learning and federated learning for pervasive services

Anastasiia Usmanova, François Portet, Philippe Lalanda, German Vega

A distillation-based approach integrating continual learning and federated learning for pervasive services

Anastasiia Usmanova[1*] , **François Portet**[2] , **Philippe Lalanda** [2] **and German Vega**[2]
[1]Grenoble INP, France
[2]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France
anastasiia.usmanova@grenoble-inp.org, Firstname.name@imag.fr

Abstract

Federated Learning, a new machine learning paradigm enhancing the use of edge devices, is receiving a lot of attention in the pervasive community to support the development of smart services. Nevertheless, this approach still needs to be adapted to the specificity of the pervasive domain. In particular, issues related to continual learning need to be addressed. In this paper, we present a distillation-based approach dealing with catastrophic forgetting in federated learning scenario. Specifically, Human Activity Recognition tasks are used as a demonstration domain.

1 Introduction

Pervasive computing promotes the integration of connected devices in our living spaces in order to assist us in our daily activities. The devices collect the data, can run some local computations and further give advises to users, act upon the environment through the services or just provide information to a global server.

We are now seeing the emergence of smarter services based on Machine Learning (ML) techniques [5]. Moreover, current implementations are actually based on distributed architectures where models are deployed and often executed in the cloud. This approach is, however, not well adapted to pervasive computing. It undergoes major limitations in terms of security, performance and cost.

Google recently proposed Federated Learning (FL) [4] [12], a new Machine Learning paradigm enhancing the use of edge devices. FL encourages the computation of local models on edge devices and sending them to a cloud server where they are aggregated into a more generic one. The new model is redistributed to devices as a bootstrap model for the next local learning iteration. FL reduces communication costs and improves security because only models are exchanged between the edge and cloud [21]. It has immediately attracted attention as a promising paradigm that can meet the challenges of ML-based pervasive applications. Nevertheless, this approach still needs to be adapted to the specificity of the pervasive domain.

In most current solutions, FL operates with static local data which stays the same during the whole training process for each client [11]. However, in real world scenarios, new data on edges is continuously available and models have to adapt to it while not forgetting the past experience. This requires the use of *Continual Learning* (CL) approaches [27]. The main challenge of CL with deep neural networks is *catastrophic forgetting* [24], [18], which is caused by optimizing the entire network according to the new tasks and not anymore to older tasks.

CL is characterized by sequential nature of the learning process, concretely by sequences of *tasks*. *Task* t is a set of classes disjoint from classes in other (previous or future) tasks, where t is its task-ID [22]. Mostly CL methods address *task-incremental learning* (task-IL) scenario [19], where information about the task-ID of examples is known at test time. However, more challenging scenario is *class-incremental learning* (class-IL), where the model has to distinguish among all the classes of all the tasks at test time [22].

In our work, we focus on a class-IL scenario of CL, which is combined with FL scenario. Our purpose was to tackle the following research questions: does FL help to prevent catastrophic forgetting on a client side? can FL help to share the past knowledge of all clients and improve their performance on unknown tasks? can we take an advantage of global server to improve the performance of clients?

We decided to use Human Activity Recognition classification (HAR) on mobile devices as our demonstration domain [25; 7] due to availability of "natural" clients: smartphones, and that this domain is not that well investigated as image classification. This choice is challenging due to the close relation of some classes (movement actions).

In this paper, we propose a distillation-based method which deals with catastrophic forgetting in Federated Continual Learning (FCL) for the HAR classification task. In section 2, we give some background on FL and CL fields and describe current existing solutions regarding the FCL problem. In section 3, we present the methodology of our work. In section 4, we propose our method. In section 5, we show experimental results and make a discussion of them. In section 5, we finish the work with a conclusion.

*This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) and by MIAI@Grenoble Alpes (ANR-19-P3IA-0003) funded by the French program Investissement d'avenir.

2 Background. Related work

2.1 Federated Learning

The main goal of conventional FL is to achieve a good global model on a server. FL process includes multiple communication rounds of: 1) local training on a client side (with static local datasets), after that clients send their updated parameters to the server; 2) parameters of gotten models are aggregated by the server to define a new global model which is further redistributed to all the clients to become an initial model in the next communication round.

FedAvg [23] uses a simple weighted average of clients' models where weights are based on the number of examples in a local dataset. FedPer [3] separates layers of clients' models on base and specialization layer and sends to the server only base layers. FedMa [28] makes a layer-wise aggregation with the fusion of similar neurons.

In our work, we want to achieve a good performance both on a server and on a clients' side, as they are the main potential users of the models. For this, we will use the FedAvg as a base method. It shows competitive result on HAR [11] and doesn't require much computational resources that is crucial for mobile devices.

2.2 Continual Learning

In standard CL on a single machine, a model iteratively learns from a sequence of tasks. At each training session, the model can have access to only one task from the sequence in order. After the training, the task is not accessible anymore.

For class-IL scenarios, CL methods can be divided into three families [22]: *regularisation-based* approaches which compute an importance of weights for previous tasks and penalise the model for changing them (EWC [18], MAS [1], PathInt [13], LwF [20]), *exemplar-based* approaches which store exemplars from previous tasks (iCarl [26], EEIL [10]), and *bias-correction* approaches which deal explicitly with bias towards recently-learned tasks (BiC [30], LUCIT [15], IL2M [6]).

2.3 Federated Continual Learning

In FCL each client has its privately accessible sequence of tasks. Each round client trains its local model on some task from its sequence and then sends its parameters o the server.

To the best of our knowledge, only few works based on a fusion of Federated and Continual Learning have been proposed so far.

For the task-IL image classification problem in FL, FedWeIT [32] was recently proposed. It is based on a decomposition of the model parameters into a dense global parameters and sparse task-adaptive which are shared between all the clients. LFedCon2 [9] and FedCL [31] deals with single-task scenario in FL. LFedCon2 uses traditional classifiers instead of DNN and propose an algorithm dealing with a concept drift based on ensemble retraining. FedCL adopt EWC [18] (regularization-based CL method which does not show the best result for the class-IL scenario [22]). FedCL aims to improve the generalization ability of federated models.

However, none of these approaches solves the problem of class-IL CL scenario in FL which we investigate in our work. That is why our research has a novel contribution in this field.

3 Methodology

3.1 Human Activity Recognition. UCI dataset

We work with a UCI HAR dataset [2] which is widely used by the HAR research community to measure and compare state-of-the-art results. The UCI dataset was collected with a Samsung Galaxy S II placed on the waist with the help of 30 volunteers. One example of data contains 128 recordings of accelerometer and gyroscope (both 3 dimensions). There are 6 classes in the dataset (number of examples per class are in brackets): 0 – Walking (1722), 1 – Walking Up (1544), 2 – Walking Down (1406), 3 – Sitting (1777), 4 – Standing (1906), 5 – Laying (1944).

To understand the relation between classes, we made a Principal Component Analysis (PCA) [17] of a UCI dataset on a last layer (before the activation) of a neural net used in our experiments (more in Section 3.3). We randomly chose 200 examples of each class for better representation and plot the first 3 principal components of this data in a 3 dimensional space (see Figure 1).

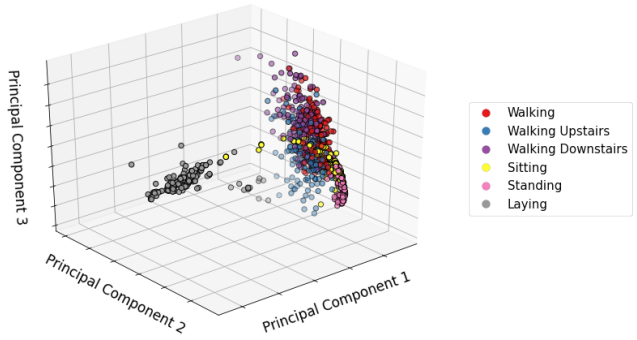


Figure 1: PCA of UCI dataset on a last layer of the used neural net, 3D space of first 3 principal components.

We can see that the class "Laying" is very distinguishable from the rest. "Sitting" and "Standing" locate close to each other. Walking movements locate together, but individually it is hard to distinguish them. Such a data distribution will have an impact on catastrophic forgetting of some classes in our experiments. For example, on Figure 2, we will see that class 5 (Laying) is never forgotten even when there is no example of this class during a training session.

3.2 Assumptions

- We focus on a **class-incremental CL scenario** since we want to classify among all the seen classes, as more common in real world settings.
- **At each communication round**, each client trains its own model on **a new local dataset** corresponding to some task, as in real world new data is collected on mobile devices in a time between communication with the server. Data from previous rounds is not available, unless otherwise mentioned.
- As we still in our work adhere the definition of a task, **any task can't be seen in a private sequence of tasks for each client more than once.** However, sets of

classes corresponding to tasks from private sequences of different clients can be overlapped.
- For the sake of simplicity, **clients behave synchronously**, so all of them take part in each communication round.

3.3 FCL problem definition

Under the assumptions above, we present following notations:

- each client $k \in \{1, 2, ..., K\}$ has its privately accessible sequence of n_k tasks $\mathcal{T}_k$:
$$\mathcal{T}_k = \left[\mathcal{T}_k^1, \mathcal{T}_k^2, ..., \mathcal{T}_k^t, ..., \mathcal{T}_k^{n_k}\right], \mathcal{T}_k^t = (C_k^t, D_k^t),$$
where $t \in \{1, ..., n_k\}$, C_k^t is a set of classes which represent the task t of a client k and $D_k^t = \{X_k^t, Y_k^t\}$ is a training data corresponding to C_k^t: $C_k^i \cap C_k^j = \varnothing$ if $i \neq j$;
- each task $\mathcal{T}_k^t$ for the client k is trained during r_k^t communication rounds and $\sum_{t=1}^{n_k} r_k^t = R$, where R is a total number of communication rounds between the server and clients;
- during communication round r client k uses training data $D_{kr} = \{X_{kr}, Y_{kr}\}$:
$$D_{kr} = D_{kr}^t \subset D_k^t, \ \sum_{t=1}^{t-1} r_k^t < r \leqslant \sum_{t=1}^{t-1} r_k^t + r_k^t,$$
where $D_{k'r'} \cap D_{k''r''} = \emptyset$, if $k' \neq k''$ and $r' \neq r''$ $(1 \leqslant k', k'' \leqslant K, 1 \leqslant r', r'' \leqslant R)$;

3.4 Clients' scenarios. Train and test sets

As the main CL scenario in FL, we observe the behaviour of a client which learns one task during half of the total number of communication rounds, and second half it learns a new task.

In notations above, **client 1** learns $n_1 = 2$ tasks in total: $\mathcal{T}_1^1 = (C_1^1, D_1^1)$ and $\mathcal{T}_1^2 = (C_1^2, D_1^2)$, where $C_1^1 = \{1\}$ and $C_1^2 = \{2\}$ – only "Walking Up" and only "Walking Down" classes, respectively; $\mathcal{T}_1 = \left[\mathcal{T}_1^1, \mathcal{T}_1^2\right]$, $r_1^1 = r_1^2 = R/2$.

For the simplicity, we assume that all other $K - 1$ clients behave similarly, and we can generalize their influence in FL process by the single **generalized client**. We also assume that all clients contain the same number of examples at each round, so the size of clients dataset will not influence forgetting. That's why, during the server aggregation the weights of generalized and the observed clients are $1/K * (K - 1)$ and $1/K$ respectively.

The generalized client performs online-learning (training on the same task each round) on data which is well-balanced and contain all the classes. So, it learns $n_g = 1$ task in total: $\mathcal{T}_g^1 = (C_g^1, D_g^1)$, where $C_g^1 = \{0, 1, 2, 3, 4, 5\}$, has its privately accessible sequence of tasks $\mathcal{T}_g = \left[\mathcal{T}_g^1\right]$, so $r_g^1 = R$.

To build train and test sets we randomly chose examples from UCI HAR dataset according to required CL scenario for each client. As mentioned above, for each client k and for each communication round r we built a **train set D_{kr} of the same size**.

We estimate the performance of the models on a **common test set** for all clients and the server. The test dataset includes 100 examples of each class (600 examples in total).

3.5 Neural Network architecture

The server and all clients use the same Neural Network architecture. As we should take into account a small processing power on mobile devices, we want to limit complexity and size of a model we use. We made a comparison of different architectures among Convolution Neural Networks (CNN) which were used in other state-of-the-art studies [16], [29].

By centralized approach, we trained the models on the dataset UCI HAR used for all the experiments in our work: 70% of data is a train set, 15% is a validation set, 15% is a test set. The models are trained using a mini-batch SGD of size 32 and a dropout rate of 0.5.

Model Architecture	Model Size	Test Acc
32-10C_256D_128D	3.3 MB	92.67
64-9C_4M_32-9C_2M_16-9C_128D	45 KB	93.76
196-16C_4M_1024D	67.8 MB	**94.64**
196-16C_4M_1024D_512D	74.1 MB	92.27

Table 1: Comparison of different model architectures on the UCI HAR test set.

Table 1 shows that a model 196-16C_4M_1024D gives the best result, and we use it in our experiments. The model includes 196 filters of a 16x1 convolution layer followed by a 1x4 max pooling layer, then by 1024 units of a dense layer and finally a softmax layer.

3.6 Settings for the experiments

All experiments were written on Python 3 using the TensorFlow 2 library and run on a CPU Intel(R) Xeon(R) 2.30GHz (2 CPU cores, 12GB available RAM).

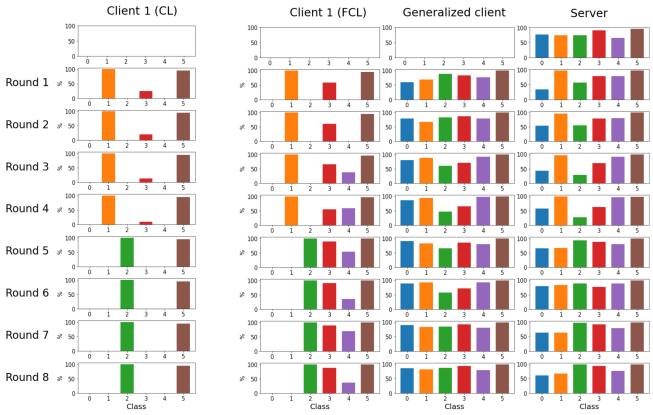
We used the same initial weights for all the models which were gotten by pre-training the chosen CNN (see Section 3.5) on a well-balanced small dataset (10 examples of each class).

We run $R = 8$ communication rounds and $E = 10$ epochs for the local training on a client side. We assumed that we have $K = 5$ clients (one client generalize the influence of $K - 1$ of them). The size of a dataset for each client k and each round r is $|D_{kr}| = 120$. We used a learning rate $\eta = 0.01$, dropout rate equal to 0.5, batch size $B = 32$ and SGD optimizer.

3.7 Demonstration of a problem

In standard CL (Figure 2(a)), we can see that class $C_1 = \{1\}$ was immediately forgotten when the learning task was changed on $C_2 = \{2\}$. But we can see always good performance on class 5 (laying), even if we don't have examples of these class during training. As we can see on Figure 1 that the laying class is very distinguishable from the others, so the initial pre-trained model successfully learnt it, and model parameters used to classify it were not changed with further training.

In FCL (Figure 2(b)), we used simple Fine-Tuning of a model when we got a new data. We still can see immediate catastrophic forgetting of the task 1 by client 1. But Federated Learning successfully transferred to the client 1 the knowledge about the other static actions (classes 3-5).



(a) CL (b) FCL

Figure 2: Demonstration of catastrophic forgetting in standard CL (a) and FCL (b).

4 Proposal

To deal with the catastrophic forgetting in FCL, we propose a distillation-based approach inspired by *Learning without Forgetting* (LwF) [20] with the use of previous model on a client side and current server model as the teachers for the present client model.

4.1 Baseline: FLwF

Knowledge distillation was originally proposed to transfer the knowledge from a large model (teacher) to a smaller one (student) [8], [14]. In LwF, authors use this technique in CL to prevent drifting while learning new tasks. For this, distillation loss was proposed to use while training the model.

First, we implemented a standard LwF method in FCL. In a standard LwF, there is one teacher model (past model of the client – round $r-1$) and one student model (current client model – round r). As we use an initial model pre-trained on all classes, each client has some knowledge about all $n=6$ classes from the beginning.

Output logits for the teacher classifier (past model of the client from a round $r-1$) is denoted in FCL as $\mathbf{o}^{r-1}(x) = \left[o_1^{r-1}(x), \ldots, o_n^{r-1}(x)\right]$, where x is an input to the network, and the output logits of the student classifier (current client model of a round r) is $\mathbf{o}^{r}(x) = [o_1^{r}(x), \ldots, o_n^{r}(x)]$.

The *distillation loss* for the client k and communication round r for LwF approach in FCL is defined as:

$$L_{dis\_cl}(D_{kr}; \theta_r^k, \theta_{r-1}^k) = \sum_{x \in X_{kr}} \sum_{i=1}^{n} -\pi_i^{r-1}(x) \log\left[\pi_i^r(x)\right],$$

where θ_r^k is the weights of the current (student) model of the client k in a communication round r (θ_{r-1}^k – previous (teacher) model), $D_{kr} = \{X_{kr}, Y_{kr}\}$ is the dataset used during communication round r by a client k, and $\pi_i^{r'}(x)$ are temperature-scaled logits of the network:

$$\pi_i^{r'}(x) = \frac{e^{o_i^{r'}(x)/T}}{\sum_{j=1}^{n} e^{o_j^{r'}(x)/T}},$$

where T is the temperature scaling parameter [14]. Temperature-scaled logits $\pi_i^{r-1}(x)$ refer to predictions of the teacher model ($\mathbf{o}^{r-1}(x)$) and $\pi_i^r(x)$ refer to predictions of the student model ($\mathbf{o}^r(x)$).

A *classification loss* (the softmax cross-entropy) in FCL is:

$$L_{class}(D_{kr}; \theta_r^k) = \sum_{(x,y) \in D_{kr}} \sum_{i=1}^{n} -y_i \log \frac{exp(o_i^r(x)}{\sum_{j=1}^{n} exp(o_j^r(x))},$$

where $(x, y) \in D_{kr} = \{X_{kr}, Y_{kr}\}$ and $D_{kr} \subset D_{kr}^t$; x is a vector of input features of a training sample, y corresponds to some class of a set C_k^t and presents as a one-hot ground truth label vector corresponding to x: $y \in \{0,1\}^{n=6}$.

The final loss in standard LwF for FCL for each client k consists of a classification loss and distillation loss computed with the current model of client k for round r and previous model of the client k for round $r-1$:

$$L_{FLwF} = \alpha L_{class} + (1-\alpha) L_{dis\_cl},$$

where α is a scalar which regularizes influence of each term.

We name this implementation as **Federated Learning without Forgetting (FLwF)**.

4.2 Proposal: FLwF-2T

We propose to take an advantage of the server which keeps a general knowledge about all the clients. Our goal is to use the server as a second teacher and to send its knowledge to a student client model. The first teacher (past model of a client) can increase the specificity performance of a student (client) which allows to be still good on tasks it has learned before. The second teacher (server) can improve general features of a client model by transferring the knowledge from all the other clients and avoid over-fitting of a client model on its new task.

We denote $\hat{\mathbf{o}}^{r-1}(x) = \left[\hat{o}_1^{r-1}(x), \ldots, \hat{o}_n^{r-1}(x)\right]$ as the output (before the activation) of the global model (server network) which was gotten after the aggregation of clients' models trained after the round $r-1$. The temperature-scaled logits of the server network are defined as follows:

$$\hat{\pi}_i^{r-1}(x) = \frac{e^{\hat{o}_i^{r-1}(x)/T}}{\sum_{j=1}^{n} e^{\hat{o}_j^{r-1}(x)/T}}.$$

The *distillation loss for the server model* is defined as:

$$L_{dis\_serv}(D_{kr}; \theta_r^k, \theta_{r-1}) = \sum_{x \in X_{kr}} \sum_{k=1}^{n} -\hat{\pi}_k^{r-1}(x) \log\left[\pi_k^r(x)\right],$$

where θ_r^k is weights of the current (student) model of the client k in a communication round r and θ_{r-1} is weights of a global model on a server gotten by aggregation of clients' models after communication round $r-1$.

The final loss for the proposed method is:

$$L_{FLwF-2T} = \alpha L_{class} + \beta L_{dis\_cl} + (1-\alpha-\beta) * L_{dis\_serv},$$

where α and β are scalars which regularize influence of terms. L_{dis_cl} refers to the distillation loss from past model (Teacher 1), defined in a Section 4.1, and L_{dis_serv} refers to a server model (Teacher 2).

For the first communication round, the model uses only the server model as a teacher.

The proposed solution helps to transfer the knowledge from the server and decrease the forgetting of previously

learnt tasks. It is a regularization-based approach as it prevents activation drift while learning new tasks. We name it **Learning without Forgetting - 2 Teachers (LwF-2T)**. The pseudo-code of FLwF-2T is presented in Algorithm 1, where loss function $L_{FLwF-2T}(...; b)$ is counted on a batch b.

Algorithm 1 FLwF-2T.

procedure SERVER EXECUTES:
  initialize server model by θ_0
  for round $r = 1, 2, ..., R$ **do**
    m = 0
    for client $k = 1, ..., K$ **in parallel do**
      $\theta_r^k \leftarrow \text{ClientUpdate}(k, r, \theta_{r-1}, \theta_{r-1}^k)$
      $m_k = |D_{kr}|$ *// Size of a dataset D_{kr}*
      $m+ = m_k$
    $\theta_r \leftarrow \sum_{k=1}^{K} \frac{m_k}{m} \theta_r^k$
function CLIENTUPDATE($k, r, \theta_{r-1}, \theta_{r-1}^k$):
  $\mathcal{B} \leftarrow$ (split D_{kr} into batches of size B)
  $\theta_r^k = \theta_{r-1}$
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $\theta_r^k \leftarrow \theta_r^k - \eta \nabla L_{FLwF-2T}(\theta, \theta_{r-1}, \theta_{r-1}^k; b)$
  return θ_r^k to the server

4.3 Metrics

To evaluate models performance, we use several metrics divided into two groups. In a first group, we use the metrics which describe FL side of a process: generality of models and performance on a specific knowledge of a client. Second group of metrics evaluate a forgetting of the learnt tasks from a CL side of a process. Let's define $a_{t,d}^{kr}$ as an accuracy of the model trained during communication round r on a task d after learning task t ($d \leqslant t$) for the client k. To compute $a_{t,d}^{kr}$ we take all the examples of classes corresponding to a task d from the test set, calculate an accuracy of the model, which was trained during communication round r, on them after the learning task t. An accuracy a_0^{kr} is calculated on the whole test set after communication round r for the client k or the server.

FL metrics:

- To evaluate generality of a model, we calculate a **general accuracy** (A_{gen}^k):
$$A_{gen}^k = \frac{1}{R} \sum_{r=1}^{R} a_0^{kr}.$$
We compute this general accuracy for the specified client, generalized client and server.
- To evaluate a performance of a model on a specific knowledge of a client, we calculate a **personal accuracy** (A_{per}^k):
$$A_{per}^k = \frac{1}{R} \sum_{r=1}^{R} a_{per}^{kr},$$
where an accuracy a_{per}^{kr} is calculated on classes which were already learnt by a client k during the rounds $1, ..., r$. We compute the personal accuracy only for for the clients with specific CL scenario.

CL metrics:

To evaluate how a model forgets previously learnt tasks, we use an average accuracy at task t on all learnt before tasks and forgetting [22]. We compute them only for the clients with specific CL scenario.

- **Average accuracy at task** t (A_t^k) calculated on all learnt before tasks for the client k:
$$A_t^k = \frac{1}{t} \sum_{d=1}^{t} a_{t,d}^k, \qquad a_{t,d}^k = a_{t,d}^k = \frac{1}{r_k^t} \sum_{r'=(r_k^t)'}^{(r_k^t)'+r_k^t} a_{t,d}^{kr'},$$
where $(r_k^t)' = \sum_{t=1}^{t-1} r_k^t$.
- **Forgetting** ($f_{t,d}^k$) shows how the model forgets the knowledge about task d after the learning a task t for the client k :
$$f_{t,d}^k = \max_{i \in \{d,...,t-1\}} \{a_{i,d}^k\} - a_{t,d}^k.$$
It can be averaged as $F_t^k = \frac{1}{t-1} \sum_{d=1}^{t-1} f_{t,d}^k$. The higher F_t^k, the more a model forgets.

5 Experiments

In this section, we present experiments where we compared different methods in FCL. We also propose to use different strategies of training for generalized client and use exemplars from learnt before tasks.

5.1 Comparison of methods

We compared following methods in FCL: Fine-Tuning in simple Continual Learning (CL-FT), Fine-Tuning in Federated Continual Learning (FCL-FT), Federated Learning without Forgetting (FLwF), our proposed method with two teachers (FLwF-2T) and Offline Learning on all training data for each client (all data). Results in a first and second parts of Tables 2, 3.

For the methods FLwF and FLwF2, we used a temperature scaling parameter $T = 2$ as commonly used in other experiments [30], [10], [22]. By using grid search we found that for FLwF $\alpha = 0.001$ and for FLwF $\alpha = 0.001$, $\beta = 0.7$ shows the best performance.

5.2 Generalized client: Fine-Tuning

In this experiment, we propose to choose a strategy of training depending on a current local data. If the local data during current communication rounds is well-balanced and contains all classes, we propose to use simple Fine-Tuning not to be dependant on a server global model which is influenced by other clients, if not, we offer to implement CL approaches to remember our past and to transfer as more available knowledge from the server as possible.

We define the strategy when Client 1 uses FLwF (FLwF-2T) and Generalized client uses FT as FLwF/FT (FLwF-2T/FT). Results in a fourth part of Tables 2, 3.

5.3 Saving the exemplars

In this experiment, we saved the exemplars of learned task in a memory. For each round, we realized the following procedure: if the task is new, we save 10 examples which are corresponded to this task in our memory; if the model has learnt this task already, we refresh out memory stock with

Method	A^1_{gen}	A^g_{gen}	A^{server}_{gen}	A^1_{per}
CL-FT	0.338	0.800	-	0.750
all data	0.715	0.899	0.865	0.977
FCL-FT	0.478	**0.794**	**0.714**	**0.750**
FLwF	0.629	0.673	0.671	0.628
FLwF-2T	**0.655**	0.679	0.680	0.629
FCL-FT + ex	0.622	0.675	0.672	0.666
FLwF + ex	0.633	0.674	0.675	0.659
FLwF-2T + ex	**0.658**	**0.683**	**0.681**	**0.664**
FLwF/FT	0.708	0.783	0.781	0.755
FLwF-2T/FT	**0.753**	**0.802**	**0.797**	**0.798**
FLwF/FT + ex	0.705	**0.789**	**0.779**	0.748
FLwF-2T/FT + ex	**0.750**	0.781	0.778	**0.755**

Table 2: **FL metrics**: general accuracy (A^k_{gen}) and personal accuracy (A^k_{per}) for Client 1 ($k = 1$), Generalized client ($k = g$) and Server ($k = server$).

Method	A^1_2	F^1_2
CL-FT	0.5	1
all data	0.981	-0.007
FCL-FT	0.5	1
FLwF	0.535	0.595
FLwF-2T	**0.578**	**0.418**
FCL-FT + ex	0.548	0.617
FLwF + ex	0.57	0.53
FLwF-2T + ex	**0.622**	**0.29**
FLwF/FT	0.696	0.392
FLwF-2T/FT	**0.76**	**0.212**
FLwF/FT + ex	0.678	0.407
FLwF-2T/FT + ex	**0.746**	**0.12**

Table 3: **CL metrics**: average accuracy at task t (A^k_t) and forgetting (F^k_t) for Client 1 ($k = 1$) and task 2 ($t = 2$).

new examples which are corresponded to this task. We implemented random sampling strategy as it shows competitive results compare to another strategies, used in class-incremental CL, and don't require much computational resources [22]. Results in a third and fifth part of Tables 2, 3.

5.4 Discussion of results

If we compare Figure 2(a) and Figure 3(b), we will see that our Continual Learning approach (FLwF-2T) helps both to increase the generality of models and to keep the knowledge from learned tasks.

First, we compared different methods in FCL for Client 1 and its specific CL scenario. **For Client 1, our method FLwF-2T shows the best general accuracy among all the methods** (Table 2). It also outperforms FLwF on a Generalized client and on the Server. Personal accuracy of FLwF-2T is decreased compared to FCL-FT due to the significant overfitting while FT. We also can see that FL with implemented CL method helps to deal with catastrophic forgetting. And our method FLwF-2T shows the best result among others in dealing with catastrophic forgetting for Client 1 (Table 3).

Then, we proposed to use different strategies of training depending on a local data for the current communication round. We can see that among compared methods, our approach with the use of FLwF-2T on a Client 1 and FT on a Generalized client significantly outperforms all other methods both in capturing generality of a model and keeping the past knowledge. So, **implementing different strategies of training for different clients depending on a local data can improve the general performance in FCL.**

Then, we made experiments with the saving of exemplars from learnt tasks. We can see, that **adding exemplars allowed to increase performance for both CL metrics for all the observed methods**. When we don't choose the training strategy for the clients, FLwF-2T + ex shows the best result among all the methods with adding exemplars. When generalized client is trained by FT, FLwF/FT + ex shows better generalization accuracy for the generalized client and the server than FLwF-2T/FT + ex.

FLwF-2T/FT – our proposed method – shows the best result among all the observed methods for both FL and CL metrics according to Tables 2, 3 (Figure 3).

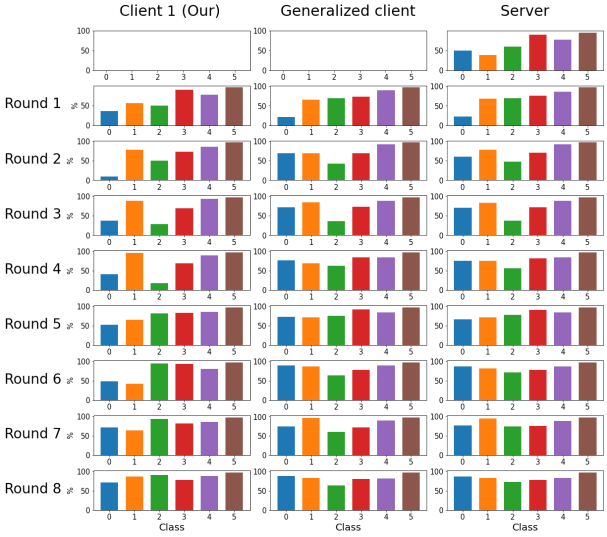


Figure 3: The best gotten result according to Tables 2, 3 shown by FLwF-2T/FT (our).

6 Conclusion

Our study presents a novel framework for the fusion of two learning paradigms: Federated Learning and Class-Incremental learning with the application in Pervasive Computing (HAR on mobile devices).

We showed that without CL approaches in FL a rapid forgetting of learnt tasks is also happened. We saw that the close relation between classes in a UCI HAR dataset influence a faster forgetting of some of them. Finally, we proposed distillation-based approach FLwF-2T for FCL, which doesn't require high computational and storage resources which is crucial for mobile devices as it uses for the training only a past model of the client and a global model sent by the server which would anyway available for the client during each communication round. We showed that it allows to increase a general knowledge for a client with specific CL scenario and to keep it's private past knowledge.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. *European Conference on Computer Vision*, 2018.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, pages 24–26, 2013.

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv:1912.00818v1*, 2019.

[4] McMahan H. B., Moore E., Ramage D., and Hampson S. Communication-efficient learning of deep networks from decentralized data. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[5] C. Becker, C. Julien, P. Lalanda, and F. Zambonelli. Pervasive computing middleware: current trends and emerging challenges. *CCF Transactions on Pervasive Computing and Interaction, vol. 1*, 2019.

[6] E. Belouadah and A. Popescu. Il2m: Class incremental learning with dual memory. *International Conference on Computer Vision,*, 2019.

[7] David Blachon, Doruk Cokun, and François Portet. On-line Context Aware Physical Activity Recognition from the Accelerometer and Audio Sensors of Smartphones. In *European Conference on Ambient Intelligence*, volume 8850 of *Ambient Intelligence*, pages 205–220, Eindhoven, Netherlands, 2014.

[8] C. Bucilua, R. Caruana, and A. Niculescu-Mizil. Model compression. *International Conference on Knowledge Discovery and Data Mining*, 2006.

[9] Fernando E. Casado, Dylan Lema, Roberto Iglesias, Carlos V. Regueiro, and Sene n Barroa. Federated and continual learning for classification tasks in a society of devices. *arXiv:2006.07129v2 [cs.LG]*, 2021.

[10] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. *European Conference on Computer Vision*, 2018.

[11] Sannara Ek, Francois Portet, Philppe Lalanda, and German Vega. A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison. In *Proceedings of the 20th International Conference on Pervasive Computing and Communications (PerCom 2022)*, 2021.

[12] Keith Bonawitz et al. Towards federated learning at scale: system design. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA, 2019.

[13] F.Zenke, B.Poole, and S.Ganguli. Continual learning through synaptic intelligence. *International Conference on Machine Learning*, 2017.

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[15] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. *International Conference on Computer Vision,*, 2019.

[16] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, pages 915–922, 2018.

[17] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc.*, 2016.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, and A. Grabska-Barwinska et al. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences, vol. 114, no. 13*, 2017.

[19] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales˘ Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv:1909.08383v2*, 2020.

[20] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40*, 2017.

[21] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials, vol. 22, no. 3, pp. 2031–2063*, 2020.

[22] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer C. Class-incremental learning: survey and performance evaluation. *arXiv:2010.15277v1*, 2020.

[23] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and et al. Communication-efficient learn- ing of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[24] McCloskey Michael and Cohen Neal J. Catastrophic interference in connectionist networks: The sequential learning problem. psychology of learning and motivation. *24. pp. 109–165. doi:10.1016/S0079-7421(08)60536-8. ISBN 978-0-12-543324-2.*, 1989.

[25] O.D.Lara and M.A.Labrador. A mobile platform for real-time human activity recognition. *IEEE Consumer Communications and Networking Conference (CCNC)*, pages 667–671, 2012.

[26] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. *Conference on Computer Vision and Pattern Recognition*, 2017.

[27] Sebastian Thrun. - A Lifelong Learning Perspective for Mobile Robot Control. In Volker Graefe, editor, *Intelligent Robots and Systems*, pages 201–214. Elsevier Science B.V., Amsterdam, 1995.

[28] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. *International Conference on Learning Representations*, 2020.

[29] Q. Wu, K. He, and X. Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 2020.

[30] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. *International Conference on Computer Vision*, 2019.

[31] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. *arXiv:2005.12657v1 [cs.LG]*, 26 May 2020.

[32] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. *arXiv:2003.03196 [cs.LG]*, 2020.