Federated Continual Learning with Adaptive Parameter Communication

Jaehong Yoon [1] [*]   Wonyong Jeong [1] [*]   Giwoong Lee [2]   Eunho Yang [1] [3]   Sung Ju Hwang [1] [3]

Abstract

There has been a surge of interest in continual learning and federated learning, both of which are important in training deep neural networks in real-world scenarios. Yet little research has been done regarding the scenario where each client learns on a sequence of tasks from private local data. This problem of *federated continual learning* poses new challenges to continual learning, such as utilizing knowledge and preventing interference from tasks learned on other clients. To resolve these issues, we propose a novel federated continual learning framework, *Federated continual learning with Adaptive Parameter Communication (APC)*, which additively decomposes the network weights into global shared parameters and sparse task-specific parameters. This decomposition allows to minimize interference between incompatible tasks, and also allows inter-client knowledge transfer by communicating the sparse task-specific parameters. Our federated continual learning framework is also communication-efficient, due to high sparsity of the parameters and sparse parameter update. We validate APC against existing federated learning and local continual learning methods under varying degree of task similarity across clients, and show that our model significantly outperforms them with large reduction in the communication cost.

1. Introduction

Continual learning or lifelong learning (Thrun, 1995) describes a learning scenario where a model continuously trains on a sequence of tasks; it is inspired by the human learning process, as a person learns to perform numerous tasks with large diversity over his/her lifespan, making use of the past knowledge to learn about new tasks without forgetting previously learned ones. Continual learning has been a long-studied topic since having such an ability leads to the potential of building a general artificial intelligence. However, there are crucial challenges in implementing it with conventional models such as deep neural networks. In early works on lifelong learning, the most important challenge was on how to perform knowledge transfer from past tasks to new ones (Kumar & Daume III, 2012; Ruvolo & Eaton, 2013). However, since the problem is relatively straightforward with deep neural networks which allow knowledge sharing through the learned networks, recent works focus on the problem of catastrophic forgetting. *Catastrophic forgetting* describes the problem where parameters or semantic representations learned for the past tasks drift to the direction of new tasks during training. The problem has been tackled by various prior work (Kirkpatrick et al., 2017; Lee et al., 2017; Shin et al., 2017; Riemer et al., 2019). More recent works tackle other issues, such as scalability or order-robustness (Schwarz et al., 2018; Yoon et al., 2020).

However, all of these models are fundamentally limited in that the models can only learn from its direct experience - it only learns from the sequence of the tasks it has trained on. Yet, humans can learn from *indirect experience* from others, through different means (e.g. verbal communications, books, or various media). Then wouldn't it be beneficial to implement such an ability to a continual learning framework, such that multiple models learning on different machines can learn from the knowledge of the asks that have been already experienced by other clients? One problem that arises here, is that due to data privacy and communication cost, it may not be possible to communicate data directly between clients or between server and the clients. Federated learning (McMahan et al., 2016) is a learning paradigm that tackles this issue by communicating the parameters instead of the raw data itself. For example, we may have a server that receives the parameters that are locally trained on multiple clients, aggregates it into a single model parameter, and sends it back to the clients. Motivated by our intuition on learning from indirect experience, we consider the problem of *Federated Continual Learning (FCL)* where we have multiple clients each of which trains on a sequence of tasks that is private to it, while communicating their parameters with a global server. We believe that obtaining a good solution to the problem is an important next step for the research of both continual learning and federated learning.

---

[*]Equal contribution   [1]KAIST, South Korea   [2]Agency for Defense Development, South Korea   [3]AITRICS, South Korea. Correspondence to: Jaehong Yoon <jaehong.yoon@kaist.ac.kr>, Wonyong Jeong <wyjeong@kaist.ac.kr>, Sung Ju Hwang <sjhwang82@kaist.ac.kr>.
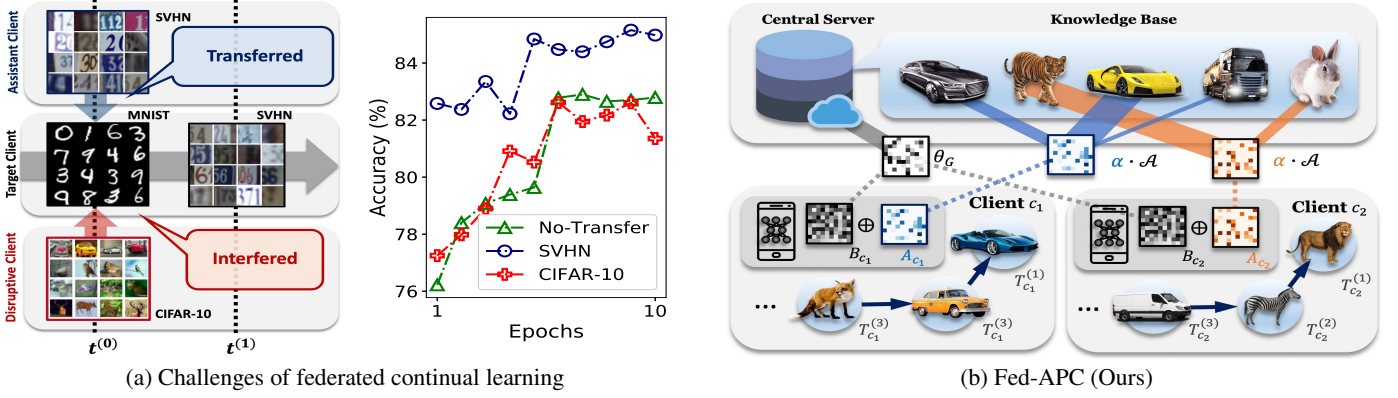
(a) Challenges of federated continual learning

(b) Fed-APC (Ours)

Figure 1. **(a) : Illustration of the effect of inter-client interference and knowledge transfer**. **(b): Overview of Fed-APC**. Each client c continuously learns on a private task sequence with inter-client knowledge transfer, while selectively utilizing the task-adaptive parameters sent from the central server, which encode knowledge of tasks learned at other clients.

Yet the problem of federated continual learning also brings new challenges. First, there is not only the catastrophic forgetting from continual learning, but also the threat of potential interference from other clients. While the use of knowledge aggregated from other clients could be helpful if they have learned on similar tasks for rapid adaptation and improving on the final performance, the knowledge from others may interfere with learning if their tasks are irrelevant. Figure 1 (a) describes this challenge with the results of a simple experiment. Here, we train a model for MNIST digit recognition by using the parameter from another client trained on a different dataset for model initialization. When the knowledge transferred from the other client is relevant to the target task (SVHN), the model starts with high accuracy, converge faster and reach higher accuracy, while the model underperforms the base model if the transferred knowledge is from a task that varies from the target task (CIFAR-10). Thus, we need to *selective utilize* knowledge from other clients to minimize the *inter-task interference* and maximize *inter-task knowledge transfer*. Another problem with the federated learning is efficient communication. We need to prevent the communication cost from becoming excessively large when utilizing the knowledge of the other clients, since in practical scenarios, the communication cost is the main bottleneck as each client may run on a device with low computing power. Thus we want the knowledge to be represented as compactly as possible.

To tackle these challenges, we propose a novel framework for federated continual learning, *Federated continual learning with Adaptive Parameter Communication (Fed-APC)*, which decomposes the model parameters into a dense global parameter and sparse task-adaptive parameters. Figure 1 (b) illustrates the overview of Fed-APC framework. Fed-APC reduces the interference between different tasks since the global parameters will encode task-generic knowledge, while the task-specific knowledge will be encoded into the task-adaptive parameters. However, we do not want to only rely on the generic knowledge, but also want the client to selectively utilize task-

specific knowledge obtained at other clients. To this end, we allow each model to take a weighted combination of the task-adaptive parameters broadcast from the server, such that it can select task-specific knowledge that is helpful for the task at hand. Fed-APC is communication-efficient, since the task-adaptive parameters are *highly sparse* and only need to be communicated once when it is created. We also perform selective update of the global shared parameters to further reduce client-to-server communication cost.

We validate our method on multiple different scenarios with varying degree of task similarity across clients against various federated learning and local continual learning models. The results show that our model obtains significantly superior performance over all baselines, adapts faster to new tasks, with largely reduced communication cost.

The main contributions of this paper are as follows:

- We introduce **a new problem of Federated Continual Learning (FCL)**, where multiple models continuously learn on distributed clients, which poses new challenges such as prevention of inter-client interference and inter-client knowledge transfer.
- We propose **a novel framework for federated continual learning**, which allows each client to adaptively update the global shared parameter and utilize the past knowledge from other clients, by communicating sparse parameters.
- We validate our model under FCL setting with both Overlapped and non-IID task sequences, on which it **largely outperforms** existing federated learning and local continual learning approaches with **significantly reduced communication cost**.

2. Related Work

Continual learning While continual learning, or lifelong learning (Thrun, 1995) is a long-studied topic with a vast

literature, we only discuss recent relevant works. A popular approach for continual learning is to use regularizations that prevent catastrophic forgetting. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) leverages Fisher Information Matrix to restrict the change of the model parameters such that the model finds solution that is good for both previous and the current task, and Lee et al. (2017) propose to learn the posterior distribution for multiple tasks as a mixture of Gaussians. While the aforementioned works consider a fixed network architecture, some researchers have proposed to prevent catastrophic forgetting by expanding the size of the networks. Rusu et al. (2016) proposes Progressive Neural Networks, which expand the networks with fixed number of neurons/filters at each layer, but the model often expands the network capacity excessively. Yoon et al. (2018) tackle this issue by expanding the networks size with minimum number of neurons/filters that are necessary via iterative neuron/filter pruning and splitting, and Xu & Zhu (2018) tackle the same problem using reinforcement learning. Yoon et al. (2020) suggest to additively decompose the parameters into shared and task-specific parameters, to minimize the increase in the network complexity by only learning the task-adaptive parameter for each new task. Another line of works allow the model to keep a small set that stores few data instances from the past tasks, which is are called *coresets*. Variational continual learning (Nguyen et al., 2018) proposes a variational framework that continuously trains the model while approximating the likelihood for the coreset, and Lopez-Paz & Ranzato (2017); Chaudhry et al. (2019) minimizes the loss on both of actual dataset and stored episodic memory. To the best of our knowledge, none of the existing approaches considered continual learning in a federated learning setting, which we tackle in this work.

Federated learning Federated Learning is a distributed learning framework under differential privacy, which aims to learn a global model on a server while aggregating the parameters learned at the clients on their private data. There are diverse approaches to aggregate the local models. FedAvg (McMahan et al., 2016) utilizes the number of data points from each client to perform weighted average of the local parameters at each iteration. TWAFL (Chen et al., 2019b) and ASO-fed (Chen et al., 2019a) follow weighted averaging in FedAvg while additionally utilizing timestemp information, newer parameters gets larger weights. In FedProx (Li et al., 2018), to tackle data and machine heterogeneity in federated learning, each client learns local parameters with proximal term which is restricting local model updates to be closer to global model but these local parameters are naively averaged in the central server. A naive averaging approach is suboptimal, since the parameters learned at clients may not be compatible due to the combinatorial nature of the distributed representations. Yurochkin et al. (2019) and Wang et al. (2020) tackle this problem by leveraging Bayesian non-parametric ap-

proaches to aggregate the model parameters in a permutation-invariant manner. Another crucial challenge in federated learning is the reduction of communication cost, as communicating the full network weights may be too costly. Chen et al. (2019b) tackle this problem by performing layer-wise parameter aggregation, where some layers(i.e. shallow layers) are aggregated in every step, but other layers(i.e. deep layers) are aggregated in last a few steps of a loop. Our method also tackles the problem of efficient communication by performing selective parameter communication.

3. Federated Continual Learning with Adaptive Parameter Communication

Motivated by the human learning process from indirect experience, we introduce continual learning under federated learning setting, which we refer to as *Federated Continual Learning (FCL)*. FCL assumes that multiple clients are trained on a sequence of tasks from private data stream, while communicating the learned parameters with a global server. In this section, we first formally define the problem, and then propose naive solutions that straightforwardly combine the existing federated learning and continual learning methods. Then, we discuss about two novel challenges that are introduced by federated continual learning, and propose a novel framework, *Adaptive Parameter Communication (APC)* which can effectively handle the two problems while also reducing the server-client communication cost.

3.1. Problem Definition

In the standard continual learning (on a single machine), the model iteratively learns from a sequence of tasks $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, ..., \mathcal{T}^{(T)}\}$ where each task $\mathcal{T}^{(t)}$ for timestep t is a labeled dataset, $\mathcal{T}^{(t)} = \{x_i^{(t)}, y_i^{(t)}\}_{i=1}^{N_t}$, which consists of N_t pairs of instances $x_i^{(t)}$ and their corresponding labels $y_i^{(t)}$. Assuming the most realistic situation, we consider the case where the task sequence is in fact a task stream with an unknown arriving order, such that the model is allowed to access $\mathcal{T}^{(t)}$ only at the given timestep t and then it cannot access it afterwards. Given $\mathcal{T}_t$ and the model learned so far, the learning objective at timestep t is as follows: $\text{minimize}_{\boldsymbol{\theta}^{(t)}} \mathcal{L}(\boldsymbol{\theta}^{(t)}; \boldsymbol{\theta}^{(t-1)}, \mathcal{T}^{(t)})$ where $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{N \times M}$ is a set of the parameters in the model at timestep t.

We now extend the conventional continual learning to the federated learning setting with multiple clients and a global server. Let us assume that we have C clients, where at each client c trains a model on a *privately accessible* sequence of tasks $\{\mathcal{T}_c^{(1)}, \mathcal{T}_c^{(2)}, ..., \mathcal{T}_c^{(t)}\} \subseteq \mathcal{T}$. Now the goal is to effectively train C continual learning models on their own private task streams, via communicating the model parameters with the global server, which aggregates the parameters sent from each client, and redistributes them to clients. In the next

section, we propose naive approaches to tackle the federated continual learning with conventional federated learning algorithms.

3.2. Communicable Continual Learning

In conventional federated learning settings, the learning is done with multiple rounds of local learning and parameter aggregation. At each round of communication t, each client $c_c \in \{c_1, \ldots, c_C\}$ and the server s perform the following two procedures: *local parameter transmission* and *parameter aggregation & broadcasting*. In the local parameter transmission step, for a randomly selected subset of clients, $\mathcal{C}^{(t)} \subseteq \{c_1, c_2, ..., c_C\}$, each client c_c sends updated parameters $\boldsymbol{\theta}_c^{(t)}$ to the server, that is obtained by training on the task $\mathcal{T}_c^{(t)}$. The update is not done at every client because some of the clients may be temporarily disconnected. Then the server aggregates the parameters $\boldsymbol{\theta}_c^{(t)}$ sent from the clients into a single parameter. The most popular frameworks for this aggregation are FedAvg (McMahan et al., 2016) and FedProx (Li et al., 2018), which we briefly describe below:

1) FedAvg (McMahan et al., 2016) aggregates the local parameters from each client by taking a weighted average of them: $\boldsymbol{\theta}_G^{(t)} \leftarrow \sum_{c=1}^{|\mathcal{C}^{(t)}|} \frac{n_c}{N} \boldsymbol{\theta}_c^{(t)}$, where N is the total number of data points at each round, and n_c is the number of data points for each client c_c. Then, each client trains on the new task t by solving the following objective: $\text{minimize}_{\boldsymbol{\theta}_c} \mathcal{L}(\boldsymbol{\theta}_c^{(t)}; \mathcal{D}_c^{(t)}, \boldsymbol{\theta}_G^{(t-1)})$.

2) FedProx (Li et al., 2018) takes a uniform average of the parameters sent from each client: $\boldsymbol{\theta}_G^{(t)} \leftarrow \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c=1}^{|\mathcal{C}^{(t)}|} \boldsymbol{\theta}_c^{(t)}$. Then, each client c_c trains on the new task as follows: $\text{minimize}_{\boldsymbol{\theta}_c} \mathcal{L}(\boldsymbol{\theta}_c^{(t)}; \mathcal{D}_c^{(t)}, \boldsymbol{\theta}_G^{(t-1)}) + \|\boldsymbol{\theta}_c^{(t)} - \boldsymbol{\theta}_G^{(t-1)}\|_2^2$, where the proximal mapping term is used to constrain the local model from deviating too much from the global parameter.

Naive training of these two models on local sequences of tasks may result in catastrophic forgetting problem. One simple solution is to use a regularization-based, such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), which allows the model to obtain a solution that is optimal for both the previous and the current tasks. There exist other advanced solutions (Rusu et al., 2016; Yoon et al., 2018; Xu & Zhu, 2018; Nguyen et al., 2018; Chaudhry et al., 2019) that successfully prevents catastrophic forgetting. However, the prevention of catastrophic forgetting at the client level is an orthogonal problem from federated learning.

Thus we focus on challenges that newly arise in this federated continual learning setting. In the federated continual learning framework, the aggregation of the parameters into a global parameter $\boldsymbol{\theta}_G$ allows *inter-client* knowledge transfer across clients, since a task $\mathcal{T}_i^{(q)}$ learned at client c_i at round q may be similar or related to $\mathcal{T}_j^{(r)}$ learned at client c_j at round r. Yet,

using a single aggregated parameter $\boldsymbol{\theta}_G$ may be suboptimal in achieving this goal, since the knowledge about all the previous tasks at one client may not be useful for others if they are irrelevant. Knowledge from irrelevant tasks may even hinder the training at each client by altering its parameters into incorrect directions, which we describe as *inter-client interference*. Another problem that is also practically important, is the communication-efficiency. Both the parameter transmission from the client to the server, and server to client will incur large communication cost, which will be problematic for the continual learning setting, since the clients may train on possibly unlimited streams of tasks.

3.3. Adaptive Parameter Communication

How can we then maximize the *knowledge transfer* between clients while minimizing the *inter-client interference*, and communication cost? We now describe our model, Federated Continual Learning with *Adaptive Parameter Communication (Fed-APC)*, which can resolve the these two problems that arise with a naive combination of continual learning approaches with federated learning framework.

The main cause of this problem, as briefly mentioned in the previous subsection, is that the knowledge of all tasks learned at multiple clients is stored into a single set of parameters $\boldsymbol{\theta}_G$. However, for the knowledge transfer to be effective, each client should *selectively* utilize only the knowledge of the *relevant* tasks that is trained at other clients. This selective transfer is also the key to minimize the inter-client interference as well as it will disregard the knowledge of irrelevant tasks that may interfere with learning.

We tackle this problem by additively decomposing the parameters, into three different types of parameters with different roles: *global parameters* that capture the global and generic knowledge across all clients, *local base parameters* that capture generic knowledge for each client, and *task-adaptive parameters* for each specific task. This additive parameter decomposition scheme is motivated by a similar method for a single machine continual learning proposed by (Yoon et al., 2020). With this additive parameter decomposition scheme, a set of the model parameters $\boldsymbol{\theta}_c^{(t)}$ for task t at client c_c can be defined as follows:

$$\boldsymbol{\theta}_c^{(t)} = \mathbf{B}_c^{(t)} \odot \mathbf{m}_c^{(t)} + \mathbf{A}_c^{(t)} + \sum_{i \in \mathcal{C}} \sum_{j=1,\ldots,t-1} \alpha_{i,j}^{(t)} \mathbf{A}_i^{(j)} \quad (1)$$

where $\mathbf{B}_c^{(t)} \in \mathbb{R}^{N \times M}$ is a base parameter for c^{th} client that is shared across all tasks, $\mathbf{m}_c^{(t)} \in \mathbb{R}^M$ is a sparse mask which enables to selectively utilize $\mathbf{B}_c^{(t)}$, $\odot$ is an element-wise multiplication, and $\mathbf{A}_c^{(t)} \in \mathbb{R}^{N \times M}$ is the highly sparse task-adaptive parameter for the task given at round t.

The first term allows selective utilization of the global knowledge. We want the base parameter $\mathbf{B}_c^{(t)}$ at each client to capture generic knowledge across all tasks across all clients. To
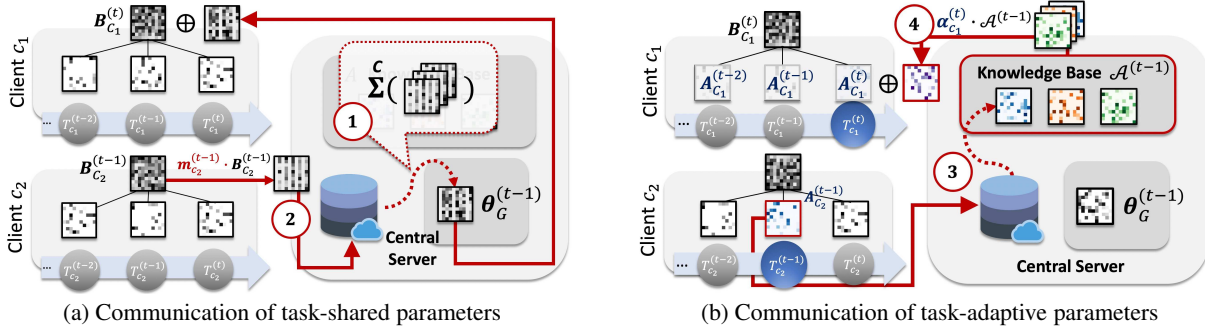
(a) Communication of task-shared parameters

(b) Communication of task-adaptive parameters

Figure 2. An illustration of *Adaptive Parameter Communication*. **(a)** A client sends sparsified local base parameter $\mathbf{B}_c$ using vector mask $\mathbf{m}_c$. After that, the server redistributes aggregated parameters of local ones $\boldsymbol{\theta}_G$ to the clients. **(b)** The knowledge base stores previous tasks-adaptive parameters of clients, and each client selectively utilizes them with a attention vector mask $(\boldsymbol{\alpha})$.

this end, we initialize it at each round t with the global parameter from the previous iteration, $\boldsymbol{\theta}_G^{(t-1)}$ which aggregates the parameters sent from the client (using either FedAvg or FedProx) (**Figure 2**(a) ,①). This allows $\mathbf{B}_c^{(t)}$ to also benefit from the *global* knowledge about all the tasks. However, since $\boldsymbol{\theta}_G^{(t-1)}$ also contains knowledge irrelevant to the current task, instead of using it as is, we learn the sparse mask $\mathbf{m}_c^{(t)}$ to select only the relevant parameters for the given task. This sparse parameter selection allows to minimize the effect of inter-client interference, and also allows for efficient communication (**Figure 2**(a) , ②).

The second term, $\mathbf{A}_c^{(t)}$ is the sparse task-adaptive parameter. Since we additively decompose the parameters, this will learn to capture knowledge about the task that is not captured by the first term, and thus will capture specific knowledge about the task $\mathcal{T}_c^{(t)}$.

The final term allows inter-client knowledge transfer. We have a set of parameters that are *transmitted* from the server, which contain all task-adaptive parameters from all the clients (**Figure 2**(b), ③). To selectively utilizes these indirect experiences from other clients, we further allocate attention on these parameters, $\boldsymbol{\alpha}_c^{(t)} \in \mathbb{R}^{|\mathcal{T}|}$, to take a weighted combination of them. By learning this attention, each client can select only the relevant task-adaptive parameters that help learn the given task (**Figure 2**(b), ④).

Training. We learn this additively decomposable parameter $\boldsymbol{\theta}_c^{(t)}$ by optimizing for the following objective:

$$\underset{\mathbf{B}_c^{(t)}, \mathbf{m}_c^{(t)}, \mathbf{A}_c^{(1:t)}, \boldsymbol{\alpha}_c^{(t)}}{\text{minimize}} \mathcal{L}\left(\boldsymbol{\theta}_c^{(t)}; \mathcal{T}_c^{(t)}\right) + \lambda_1 \Omega(\{\mathbf{m}_c^{(t)}, \mathbf{A}_c^{(1:t)}\})$$
$$+ \lambda_2 \sum_{i=1}^{t-1} \|\Delta\mathbf{B}_c^{(t)} \odot \mathbf{m}_c^{(i)} - \Delta\mathbf{A}_c^{(i)}\|_2^2, \quad (2)$$

where $\mathcal{L}$ is a loss function and $\Omega(\cdot)$ is a sparsity-inducing regularization term for the task adaptive parameter and the masking variable (we use ℓ_1-norm regularization), to make them

sparse. The final regularization term is used for retroactive update of the past task-adaptive parameters, which helps the task-adaptive parameters to maintain the original solutions for the target tasks, by reflecting the change of the base parameter. Here, $\Delta\mathbf{B}_c^{(t)} = \mathbf{B}_c^{(t)} - \mathbf{B}_c^{(t-1)}$ is the difference between the base parameter at the current and previous timestep, and $\Delta\mathbf{A}_c^{(i)}$ is the difference between the task-adaptive parameter for task i at the current and previous timestep. This regularization is essential for preventing catastrophic forgetting. λ_1 and λ_2 are hyperparameters controlling the effect of the two regularizers.

3.4. Efficient communication via sparse parameters

Fed-APC learns via server-to-client communication. As discussed earlier, a crucial challenge here is to reduce the communication cost. We describe what happens at the client and the server at each step.

Client: At the beginning of each training step, each client c updates the base parameter by nonzero components of the global parameter sent from the server; that is, $\mathbf{B}_c(i) = \boldsymbol{\theta}_G(i)$ where i is a nonzero element of the global parameter. After training the model using Eq. (2), it obtains a sparsified base parameter $\hat{\mathbf{B}}_c^{(t)} = \mathbf{B}_c^{(t)} \odot \mathbf{m}_c^{(t)}$ and the newly learned task-adaptive parameter $\mathbf{A}_c^{(t)}$. Then, the client sends both $\hat{\mathbf{B}}_c^{(t)}$ and $\mathbf{A}_c^{(t)}$ to the server. Since both parameters are highly sparse, this results in the large reduction of the client-to-server communication cost.

Server: The central server first aggregates the *sparsified* base parameters sent from all the clients by taking an weighted average of them: $\boldsymbol{\theta}_G^{(t)} = \frac{1}{C} \sum_c \hat{\mathbf{B}}_c^{(t)}$. Then, it broadcasts $\boldsymbol{\theta}_G^{(t)}$ along with all task adaptive parameters $\mathcal{A}^{(t)} = \{\mathbf{A}_c^{(t)}, \ldots, \mathbf{A}_C^{(t)}\}$ to all the clients. The full algorithm for our federated learning with adaptive parameter communication is given in Algorithm 1.

Algorithm 1 Fed-APC

---

input Dataset $\{\mathcal{D}_c^{(1:t)}\}_{c=1}^C$, and Global Parameter $\boldsymbol{\theta}_G^{(0)}$
output $\{\mathbf{B}_c, \mathbf{m}_c^{(1:t)}, \boldsymbol{\alpha}_c^{(1:t)}, \mathbf{A}_c^{(1:t)}\}_{c=1}^C$

1: **function** RunCentralServer
2:    UpdateCLClient(c, $\boldsymbol{\theta}_G^{(0)}$, $\boldsymbol{\phi}$) for all $c \in \{1, ..., C\}$
3:    **for** round $r = 1, 2, ...$ **do**
4:       Parameters are transferred from $\mathcal{C}^{(r-1)} \subseteq \{1, ..., C\}$
5:       Compute $\boldsymbol{\theta}_G^{(r-1)} \leftarrow \frac{1}{|\mathcal{C}^{(r-1)}|} \sum_{c \in \mathcal{C}^{(r-1)}} \hat{\mathbf{B}}_c^{(r-1)}$
6:       **for** each client $c \in \mathcal{C}^{(r-1)}$ **in parallel do**
7:          UpdateCLClient(c, $\boldsymbol{\theta}_G^{(r-1)}$, $\mathcal{A}_\mathcal{C}^{(r-1)}$)
8:       **end for**
9:    **end for**
10: **end function**
11:
12: **function** UpdateCLClient(c, $\boldsymbol{\theta}_G^{(t-1)}$, $\mathcal{A}^{(t-1)}$)
13:    **if** new task t arrives **then**
14:       Initialize task-adaptive Parameters $\{\mathbf{m}_c^{(t)}, \mathbf{A}_c^{(t)}, \boldsymbol{\alpha}_c^{(t)}\}$
15:    **end if**
16:    Minimize Eq. (2) to update $\mathbf{B}_c^{(t)}$, $\boldsymbol{\alpha}_c^{(t)}$, and $\{\mathbf{A}_c^{(i)}\}_{i=1}^t$
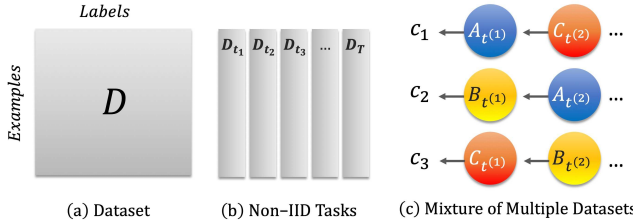17: **end function**

Figure 3. **Configuration of task sequences:** We first split a dataset D into multiple sub-tasks D_T in non-IID manner ((a) and (b)). Then we distribute them to multiple clients (denoted as C). Mixed tasks from multiple datasets (illustrated as colored circles) are distributed across all clients (c).

4. Experiments

We now validate Fed-APC under different configurations of task sequences against relevant baselines.

4.1. Tasks and Baselines

Task configuration We verify our methods Fed-APC under two different sets of task sequences, which are namely Overlapped-CIFAR-100 and NonIID-50.

1) Overlapped-CIFAR-100: We group 100 classes of CIFAR-100 dataset into 20 superclasses, and create 20 non-iid tasks, each of which is a classification of five classes in each superclass that are disjoint from the classes used by other tasks. Then, we randomly sample 10 tasks out of 20 tasks and split instances to create a task sequence for each of the 5 clients with overlapping tasks.

2) NonIID-50: We use the following eight benchmark datasets: MNIST (LeCun et al., 1998), CIFAR-10 & CIFAR-100 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), Fashion-MNIST (Xiao et al., 2017), Not-MNIST (Bulatov, 2011), FaceScrub (Ng & Winkler, 2014),

and TrafficSigns (Stallkamp et al., 2011). The number of classes for each dataset ranges from 10 to 100. We pre-process the datasets to set the shape of all the images to $32 \times 32 \times 3$. We split the classes in the eight dataset into 50 non-IID tasks, each of which is composed of 5 classes that are disjoint from the classes used for the other tasks. Specifically, we generate 2 non-IID tasks from the dataset with 10 classes. For datasets with 100 classes, we generate the following number of tasks for each dataset: 15 for CIFAR-100, and 16 for FaceScrub. We discard the remaining classes. After generating and processing tasks, we randomly distribute them to multiple clients as illustrated in Figure 3 (c).

Experimetal setup We use a modified version of LeNet (LeCun et al., 1998) for the experiments with both Overlapped-CIFAR-100 and NonIID-50 dataset. As for other experimental setups, we followed the settings from Serrà et al. (2018) and Yoon et al. (2020). For detailed descriptions of the task configuration, network architecture and hyperparameters used, please see **supplementary file**. We will release the codes upon the acceptance of our paper.

Baselines and our models **1) L2T**: A simple continual learning model with the ℓ_2-transfer regularizer $\lambda \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\|_2^2$ when training for task t, to alleviate catastrophic forgetting. **2) EWC**: Elastic Weight Consolidation (Kirkpatrick et al., 2017), which adopts a regularizer based on Fisher information matrix to alleviate forgetting. **3) APD**: Continual learning with Additive Parameter Decomposition (Yoon et al., 2020), which allows to additively decompose parameters to prevent catastrophic forgetting. **4) Fed-L2T**: Federated continual learning, that is trained using either *FedAvg*, *FedProx* algorithm with ℓ_2-transfer regularizer. **5) Fed-EWC**: Federated continual leaning with EWC. **6) Fed-APD**: Federated continual learning with APD, where each client sends the task-shared parameters to the central server which is aggregated by federated learning algorithms. **7) Fed-APC**: Our Adaptive Parameter Communication which alleviates inter-task interference and promotes inter-client knowledge transfer, with efficient parameter communication.

4.2. Experimental results

We first validate our model on both CIFAR-100 and NonIID-50 task sequences against naive federated continual learning baselines. Table 1 shows the final average per-task performance after the completion of (federated) continual learning. We observe that Federated continual learning (FCL) approaches with FedAvg degenerate the performance of continual learning methods over the same methods without federated learning. This is because the aggregation of all client parameters that are learned on irrelevant tasks results in severe interference in the learning for each task, that may lead to catastrophic forgetting and suboptimal task adaptation. Although FedProx-based continual learning methods obtain bet-

Methods	Overlapped-CIFAR-100			NonIID-50		
	Accuracy (%)	Capacity (MB)	Cost(C) (MB)	Accuracy (%)	Capacity (%)	Cost(C) (MB)
STL	57.15 ± 0.07	121.5 (1,000%)	N/A	85.78 ± 0.17	121.5 (1,000%)	N/A
L2T	44.43 ± 0.23	12.2 (100%)	N/A	71.49 ± 0.75	12.2 (100%)	N/A
EWC	44.60 ± 0.43	12.2 (100%)	N/A	74.30 ± 0.08	12.2 (100%)	N/A
APD	50.90 ± 0.33	14.5 (119%)	N/A	82.41 ± 0.63	17.6 (145%)	N/A
FedAvg-L2T	38.20 ± 0.36	12.2 (100%)	12.1 (100%)	58.09 ± 1.10	12.2 (100%)	12.1 (100%)
FedAvg-EWC	41.34 ± 0.05	12.2 (100%)	12.1 (100%)	65.11 ± 1.15	12.2 (100%)	12.1 (100%)
FedAvg-APD	51.94 ± 0.16	15.1 (124%)	12.1 (100%)	79.55 ± 0.35	17.1 (141%)	12.1 (100%)
FedProx-L2T	38.69 ± 0.30	12.2 (100%)	12.1 (100%)	65.66 ± 1.24	12.2 (100%)	12.1 (100%)
FedProx-EWC	41.98 ± 0.47	12.2 (100%)	12.1 (100%)	68.18 ± 0.58	12.2 (100%)	12.1 (100%)
FedProx-APD	52.69 ± 0.41	15.1 (124%)	12.1 (100%)	82.95 ± 1.01	16.2 (133%)	12.1 (100%)
Fed-APC (Ours)	$\mathbf{54.70 \pm 0.24}$	**14.8 (122%)**	12.4 (102%)	$\mathbf{84.71 \pm 0.69}$	**14.8 (122%)**	12.4 (102%)
Fed-APC (Ours)	$\mathbf{55.16 \pm 0.19}$	**15.3 (126%)**	**4.0 (33%)**	$\mathbf{84.11 \pm 0.27}$	**15.6 (128%)**	**4.0 (33%)**

Table 1. Averaged Per-task performance on *Overlapped-CIFAR-100* and *NonIID-50* on (federated) continual learning with 5 clients. We measured task average accuracy and model capacity ratio after completing all learning phases over 3 individual trials.
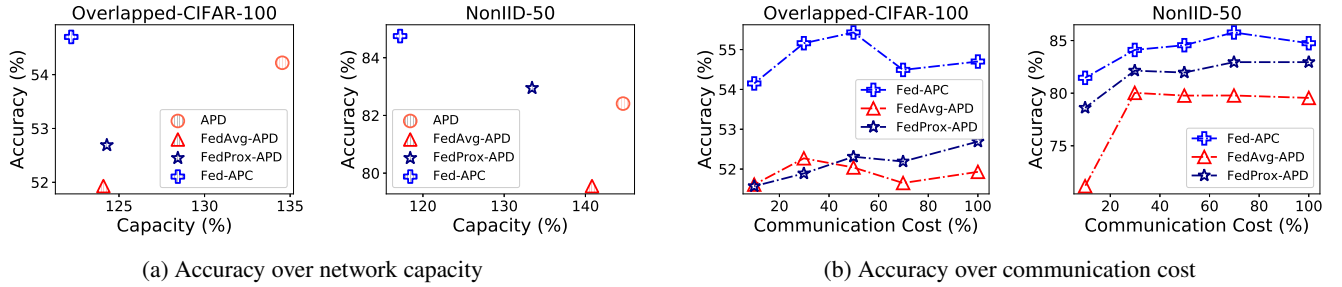

(a) Accuracy over network capacity   (b) Accuracy over communication cost

Figure 4. **(a) Accuracy over network capacity.** We report the number of parameters used for each method compared to the original network. **(b) Accuracy over communication cost.** We report the relative communication cost to the original network. All experimental results are averaged over the 5 clients used for the experiments, over 3 independent trials.

ter performance over FedAvg-based methods, they still suffer from the inter-client interference. On the other hand our Fed-APC significantly outperforms baselines, including single-machine continual learning (CL) methods. The performance gain is greater on Overlapped-CIFAR-100, where there is a chance that the same tasks (with disjoint instances) can be experienced by multiple clients than on NonIID-50, where all tasks are disjoint.

We also report accuracy over network capacity in Table 1 and Figure 4(a), which we measure by the number of parameters used. We observe that Fed-APC obtains much higher accuracy while utilizing less number of parameters compared to FedAvg-APD and FedProx-APD. This efficiency mainly comes from the reuse of task-adaptive parameters from other clients, which is not possible with single-machine CL method or naive FCL methods.

We further examine the communication cost of each method. Table 1 shows the client-to-server communication cost (Cost(C)). Further, Figure 4(b) shows the accuracy as function of communication cost. We observe that Fed-APC is significantly more communication-efficient than FedAvg-APD and FedProx-APD, even though it broadcasts task-adaptive parameters, due to high sparsity of the parameters.

Effect of Inter-client Knowledge Transfer The plots in top row of Figure 5 show how the model accuracy changes during training, for 3rd, 5th, 6th, and 8th tasks of NonIID-50 task sequences. We observe that while federated continual learning baselines (FedAvg-APD and FedProx-APD) suffer from inter-client interference on others. Contrarily, our model, Fed-APC mostly outperforms APD on all tasks, due to its ability to selectively transfer knowledge from both the global parameter, and the task-adaptive parameters of other clients. We also see that for later tasks, Fed-APC starts at significantly higher accuracy at initialization, which is another advantage of inter-client knowledge transfer.

Catastrophic forgetting Further, we examine how the performance of the past tasks change during continual learning of our model and the FCL baselines, to see the severity of **catastrophic forgetting** with each method. The bottom row of Figure 5 shows the performance of Fed-APC and FCL baselines on the same tasks, at the end of training for each task. We observe that naive FCL baselines sometimes suffer from more severe catastrophic forgetting than EWC because of *inter-client interference*, where the knowledge of irrelevant tasks from other clients overwrites the knowledge of the past tasks at each task. Contrarily, our model shows no sign of catastrophic forgetting. This is mainly due to the additive parameter decomposition and selective utilization of the global/task-adaptive parameters, which allows it to effectively alleviate *inter-client*
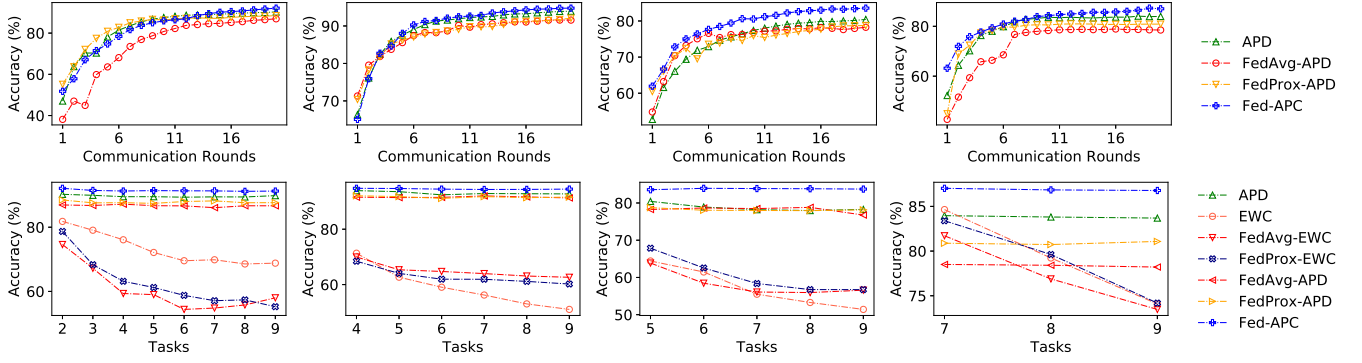
Figure 5. **Top**: Performance comparison about current task adaptation at 3_{rd}, 5_{th}, 6_{th}, and 8_{th} tasks during federated continual learning on *NonIID-50*. **Bottom**: Analysis of Catastrophic Forgetting on the same tasks. All performance are averaged on clients.

Methods	NonIID-50			
	Accuracy	**Capa.**	**Cost(C)**	**Cost(S)**
Fed-APC	**84.11%**	**128%**	**33%**	**96%**
w/o $\mathbf{B}_c^{(t)}$ comm.	77.88%	115%	2%	8%
w/o $\mathbf{A}_c^{(t)}$ comm.	79.21%	130%	30%	83%
w/o $\mathbf{A}_c^{(t)}$	65.66%	100%	30%	83%
w/o $\mathbf{m}_c^{(t)}$	78.71%	143%	104%	121%

Table 2. Ablation studies to analyze the effectiveness of parameter decomposition on Adaptive Parameter Communication. All experiments performed on NonIID-50 dataset.

interference. FedAvg-APD or FedProx-APD also do not suffer from catastrophic forgetting as they also decompose parameters, but they yield inferior performance due to ineffective knowledge transfer.

Ablation study We perform an ablation study to analyze the role of each component for further understanding of our method. We compare the performance of four different variations of our model. **w/o $\mathbf{B}_c^{(t)}$ communication** describes the model that does not transfer the base parameter $\mathbf{B}_c^{(t)}$ and only communicates task-adaptive ones. **w/o $\mathbf{A}_c^{(t)}$ communication** is a model that does not communicate task-adaptive parameters. **w/o $\mathbf{A}_c^{(t)}$** is the model which trains the model only with sparse transmission of local base parameter, and **w/o $\mathbf{m}_c^{(t)}$** is the model without the sparse vector mask. As shown in Table 2, without communicating $\mathbf{B}_c^{(t)}$ or $\mathbf{A}_c^{(t)}$, the model yields significantly lower performance compared to the full model since they do not benefit from inter-task knowledge transfer. The model **w/o $\mathbf{A}_c^{(t)}$** obtains very low performance due to catastrophic forgetting, and the model **w/o $\mathbf{m}_c^{(t)}$** the sparse mask achieves lower accuracy with larger capacity and communication cost, which demonstrates the importance of performing selective transfer.

Inter-task Knowledge Transfer By analyzing the attention α_{ij} in Eq. (1), we can examine which task parameters from other clients each client selected. Figure 6, shows example of the attention weights that are learned for the 0th split of *MNIST* and 10th split of *CIFAR-100*. We observe that large attentions are allocated to the task parameters from the
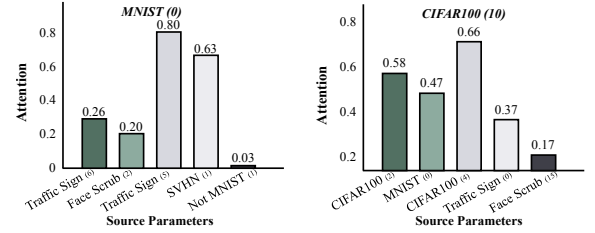

Figure 6. Inter-task knowledge transfer using task-adaptive parameters in Fed-APC for NonIID-50. We compare the scale of the attentions at first FC layer which gives the weights on the source task-adaptive parameters from other clients.

same dataset (CIFAR-100 utilizes parameters from CIFAR-100 tasks with disjoint classes), or from a similar dataset (MNIST utilizes parameters from Traffic Sign and SVHN). This shows that Fed-APC effectively selects beneficial parameters to maximize *inter-client* knowledge transfer. This is an impressive result since it does not know which datasets the parameters are trained on.

5. Conclusion

We tackled a novel problem of federated continual learning, whose goal is to continuously learn local models at each client while allowing it to utilize indirect experience (task knowledge) from other clients. This poses new challenges such as inter-client knowledge transfer and prevention of inter-client interference. To tackle these challenges, we additively decompose the model parameters at each client into the global shared parameter that is shared across all clients, and sparse local task-adaptive parameters that are specific to each task. Further, we allowed each model to selectively update the global task-shared parameters and selectively utilize the task-adaptive parameters from other clients. The experimental validation of our model under various task similarity across clients, against existing federated learning and continual learning baselines shows that our model obtains significantly higher accuracy with reduced communication cost. We believe that federated continual learning is a practically important topic of large interests to both research communities of CL and FL, that will lead to new research directions.

References

Bulatov, Y. Not-mnist dataset. 2011.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Chen, Y., Ning, Y., and Rangwala, H. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134*, 2019a.

Chen, Y., Sun, X., and Jin, Y. Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation. *arXiv preprint arXiv:1903.07424*, 2019b.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pp. 201611835, 2017.

Krizhevsky, A. and Hinton, G. E. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.

Kumar, A. and Daume III, H. Learning task grouping and overlap in multi-task learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Ng, H.-W. and Winkler, S. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pp. 343–347. IEEE, 2014.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Rusu, A., Rabinowitz, N., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Ruvolo, P. and Eaton, E. Ella: An efficient lifelong learning algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, 2011.

Thrun, S. *A Lifelong Learning Perspective for Mobile Robot Control.* Elsevier, 1995.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkluqlSFDS.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, J. and Zhu, Z. Reinforced continual learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Yoon, J., Kim, S., Yang, E., and Hwang, S. J. Scalable and order-robust continual learning with additive parameter decomposition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, T. N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Federated Continual Learning with Adaptive Parameter Communication: Supplementary File

6. Experimental Details

We provide details of the experimental settings. Also, we additionally report experimental results including quantitative analysis and ablation study.

6.1. Network Architecture

We utilize a modified version of LeNet as our base architecture for all baselines and proposed models. The first two layers are convolutional neural layers of 20 and 50 filters with the same 5×5 kernel sizes followed by the two fully-connected layers of 800 and 500 units each. Rectified linear units activations and local response normalization are subsequently applied for each layers. We use 2×2 max-pooling after each convolutional layer. Fully-connected layers with softmax outputs are utilized as our final layers. All layers are initialized based on the varaiance scaling method. Detailed description of the architecture is described in Table 3.

Layer	Filter Shape	Stride	Output
Input	N/A	N/A	$32 \times 32 \times 3$
Conv 1	$5 \times 5 \times 20$	1	$32 \times 32 \times 20$
Max Pooling 1	3×3	2	$16 \times 16 \times 20$
Conv 2	$5 \times 5 \times 50$	1	$16 \times 16 \times 50$
Max Pooling 2	3×3	2	$8 \times 8 \times 50$
Flatten	3200	N/A	$1 \times 1 \times 3200$
FC 1	800	N/A	$1 \times 1 \times 800$
FC 2	500	N/A	$1 \times 1 \times 500$
Softmax	Classifier	N/A	$1 \times 1 \times 5 \times T$
Total Number of Parameters			3,012,920

Table 3. Base Network Architecture and Total Number of Parameters of both Fed-APC and All Baseline Models. T describes the number of arrived tasks in continual learning.

We use Adam optimizer with adaptive learning rate decay, which decays learning rate by a factor of 3 for every 5 epochs that validation loss does not consecutively decrease. We stop training in advance and start training the next task (if available) when the learning rate reaches $1e^{-7}$, which is initialized by $1e^{-3} \times \frac{1}{3}$ at the beginning of each new task. For experiments with 5 clients, we set 100 for minibatch size, 20 for rounds per task, 1 for an epoch per round, and 1.0 for client fraction per round. In a case of experiments with 20 and 100 clients, we set the same settings except reducing minibatch size from 100 to 10 and exploring client fraction 0.25 and 0.5, respectively. In terms of hyperparameters of our models, we set $\lambda_1 = [1e^{-1}, 4e^{-1}]$ and $\lambda_2 = 100$ for all experiments.

6.2. Dataset

We create both Overlapped-CIFAR-100 and NonIID-50 datasets and detailed information is described in Table 4. For Overlapped-CIFAR-100, we generate 20 non-iid tasks based on 20 superclasses, which hold 5 subclasses. We split instances of 20 tasks according to the number of clients (5, 20, and 100) and then distribute the tasks across all clients. The average performance of single task learning on the dataset is $57.15 \pm 0.07(\%)$, measured by our base architecture which described in section 6.1.

Overlapped-CIFAR-100			
Dataset	**# Classes**	**# Tasks**	**# Classes per Task**
CIFAR-100	100	20	5
Total	100	20	100

Table 4. Detailed configuration of *Overlapped-CIFAR-100*.

For NonIID-50 dataset, we utilize 8 heterogenous datasets and create 50 non-iid tasks in total as shown in Table 5. Then we arbitrarily select 10 tasks without duplication and distribtue them to 5 clients. The average performance of single task learning on the dataset is $85.78 \pm 0.17(\%)$, measured by our base architecture which described in section 6.1.

NonIID-50			
Dataset	**# Classes**	**# Tasks**	**# Classes per Task**
CIFAR-100	100	15	5
Face Scrub	100	16	5
Traffic Signs	43	9	5 (3)
SVHN	10	2	5
MNIST	10	2	5
CIFAR-10	10	2	5
Not MNIST	10	2	5
Fashion MNIST	10	2	5
Total	293	50	248

Table 5. Detailed configuration of *NonIID-50*.

7. Additional Ablation Study

We provide the results of further experimental results of our model: 1) effect of the frequency of the communication, by the number of epochs per round, and 2) effect of the number of clients.

7.1. Effect of the Communication Frequency

We provide an analysis about the effect of the communication frequency on the performance of the model, measured
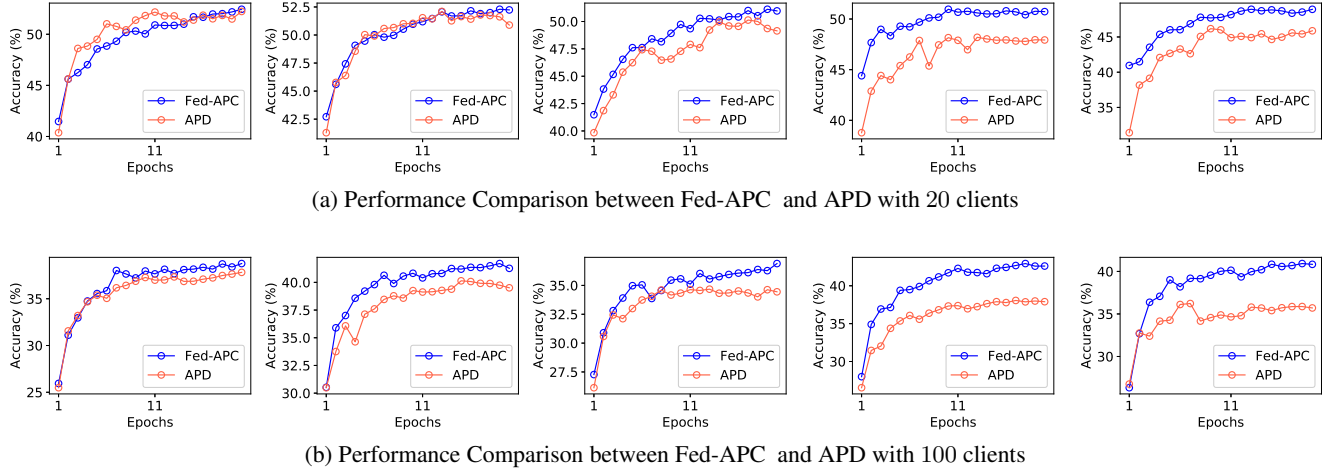
(a) Performance Comparison between Fed-APC and APD with 20 clients

(b) Performance Comparison between Fed-APC and APD with 100 clients

Figure 7. (a) Task adaptation comparison between Fed-APC and APD with 20 clients in federated continual learning scenario (b) Task adaptation comparison between Fed-APC and APD with 100 clients in federated continual learning scenario. Both of them visualize the last 5 tasks out of 10 tasks per client. *Overlapped-CIFAR-100* dataset are used after splitting instances according to the number of clients (20 and 100).
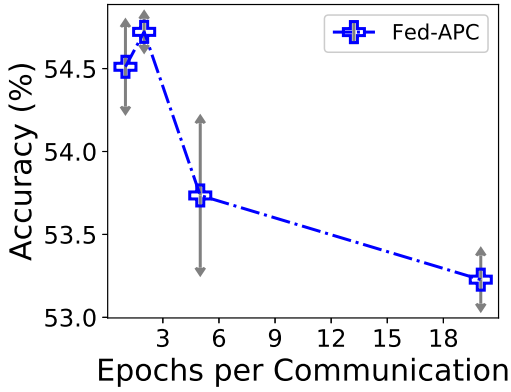

Figure 8. Average Per-task Performance with error bars across the number of training epochs per communication rounds on *Overlapped-CIFAR-100* for Fed-APC with 5 clients. All models transmit full of local base parameters and highly sparse task-adaptive parameters. All results are the mean accuracies over 5 clients and we run 3 random splits. Gray arrows at each point describes the error bar about the standard deviation of the performance.

by the number of training epochs per communication round. We run the 4 different Fed-APC given $1, 2, 5,$ and 20 training epochs per round. Figure 8 and Table 6 shows the performance of our Fed-APC variants. As clients frequently update the model parameters through the communication with the central server, the model gets higher performance while maintaining smaller network capacity since the model with a frequent communication efficiently updates the model parameters as transferring the inter-client knowledge. However, it requires much heavier communication costs than the model with sparser communication. For example, the model who trains 1 epochs at each round may need to about 16.9 times

	Overlapped-CIFAR-100		
Methods	**Accuracy (%)**	**Capacity**	**Epochs / Round**
Fed-APC (Ours)	**54.70 ± 0.24**	**14.8 MB (122%)**	1
Fed-APC (Ours)	**54.72 ± 0.08**	15.3 MB (126%)	2
Fed-APC (Ours)	53.73 ± 0.44	16.5 MB (136%)	5
Fed-APC (Ours)	53.22± 0.14	17.5 MB (144%)	20

Table 6. Average Per-task performance on *Overlapped-CIFAR-100* for Fed-APC with 5 clients. All results are the mean accuracies over 5 clients and we run 3 random splits and we include standard deviations for all experiments.

larger entire communication cost than the model who trains 20 epochs at each round. Hence, there is a trade-off between model performance of federated continual learning and communication efficiency whereas Fed-APC variants consistently outperform (federated) continual learning baselines.

	20 clients		100 clients	
Methods	**Acc.(%)**	**Capa.**	**Acc.(%)**	**Capa.**
APD	46.48%	153%	37.50%	329%
Fed-APC (Ours)	**50.38%**	155%	**39.58%**	330%

Table 7. Comparison of averaged per-task performance and network capacity between APD and Fed-APC on *Overlapped-CIFAR-100* when 20 and 100 clients are participated.

7.2. Effect of The Number of Clients

We further analyze what happens when our model communicates with larger number of clients. In this setting, the central server randomly selects 5 clients out of 20 or 100 clients, respectively, at each round. We set client fraction as 0.25 for experiments with 20 clients and 0.05 for experiments with 100 clients. The selected clients communicate with the central server to share their parameters while the other clients which are not chosen simply learn their task without sharing their weights. When clients start learning new task, however, the central server broadcasts the global knowledge to all clients regardless of clients selection.

To experiment task adaptation with the larger number of clients in the setting mentioned above, we compare our model Fed-APC with a strong baseline in continual learning, Additive Parameter Decomposition (APD). We have 20 and 100 independent APDs trained on the same tasks in the identical order with that of 20 and 100 Fed-APC experiments. Each baseline model independently trains without any communicating with the central server and transferring knowledge to the other clients. For this experiement, we use *Overlapped-CIFAR-100* dataset. We also split instances per task according to the number of clients, for both 20 and 100-clients cases, to preserve the instance-level independence across all clients.

Figure 7 shows the adaptation plots of the last 5 tasks for Fed-APC and APD with 20 and 100 clients, respectively ((a) and (b)). We observe that our Fed-APC achieves significant performance gains and adapts more rapidly compared to the baseline which do not perform inter-client knowledge transfer. In both cases where 20 and 100 clients are used, the results clearly demonstrate that each local clients in Fed-APC successfully utilize inter-client knowledge by aggregating their local base parameters and selectively transmitting their previous task-adaptive parameters.

Table 7 shows averaged per-task performance and network capacity of both Fed-APC and APD with 20 and 100 clients. All results are the mean accuracy over all clients. The table also clearly shows that our Fed-APC outperforms the baseline models. For the experiments with 20 clients, Fed-APC shows $3.90\%p$ better performance over APD in similar level of network capacity. For experiments with 100 clients, Fed-APC performs $2.08\%p$ better than baseline model in the same level of network capacity.