Addressing Client Drift in Federated Continual Learning with Adaptive Optimization

Yeshwanth Venkatesha, Youngeun Kim, Hyoungseob Park, Yuhang Li, and Priyadarshini Panda
Yale University, New Haven CT 06511, USA
{yeshwanth.venkatesha, youngeun.kim, hyoungseob.park, yuhang.li, priya.panda}@yale.edu

Abstract. Federated learning has been extensively studied and is the prevalent method for privacy-preserving distributed learning in edge devices. Correspondingly, continual learning is an emerging field targeted towards learning multiple tasks sequentially. However, there is little attention towards additional challenges emerging when federated aggregation is performed in a continual learning system. We identify client drift as one of the key weaknesses that arises when vanilla federated averaging is applied in such a system, especially since each client can independently have different order of tasks. We outline a framework for performing Federated Continual Learning (FCL) by using NetTailor as a candidate continual learning approach and show the extent of the problem of client drift. We show that adaptive federated optimization can reduce the adverse impact of client drift and showcase its effectiveness on CIFAR100, MiniImagenet, and Decathlon benchmarks. Further, we provide an empirical analysis highlighting the interplay between different hyperparameters such as client and server learning rates, the number of local training iterations, and communication rounds. Finally, we evaluate our framework on useful characteristics of federated learning systems such as scalability, robustness to the skewness in clients’ data distribution, and stragglers.

Keywords: Federated Learning, Continual Learning, Client Drift, Adaptive Optimization

1 Introduction

Federated learning has drawn widespread attention to utilizing millions of edge devices that accumulate massive amounts of data while preserving data privacy. Since these devices constantly generate enormous amounts of data, we need models that adapt to the trends in the new data. Several studies have addressed this problem of learning on a sequential stream of data termed continual learning or lifelong learning. While both federated learning and continual learning have been widely studied independently in recent years, the effect of federated aggregation in a continual learning system, referred to as
federated continual learning (FCL) has not been sufficiently explored despite its relevance to practical scenarios.

In an FCL system, the challenges are twofold. First, following the popular formulation of continual learning as a sequence of tasks, the challenge is to learn new tasks while avoiding the overwriting of previously learned tasks, commonly referred to as Catastrophic Forgetting [11,19]. Second, similar to catastrophic forgetting between tasks in continual learning, federated learning handles heterogeneous data distributions from different clients resulting in a phenomenon called client drift [17]. As illustrated in Fig. 1a, when the data is heterogeneous, each of the clients pushes the model in a different direction in the optimization space and takes the model away from the true optima resulting in client drift. Further, since there are multiple clients, the relative order of tasks in each client determines the overall client drift. We describe three practical cases with respect to the ordering of tasks. First, when all the clients contain the data for all the tasks simultaneously, defining the scenario of federated multi-task learning (FMTL) which has been explored previously [45,7,29]. Second, the ideal case of federated continual learning where all the clients synchronously train on the same sequence of tasks. Finally, the case where each client independently trains on a different sequence of tasks. Fig. 1b visually depicts the aforementioned order of tasks. Note that for simplicity we assume every client contains the dataset for all the tasks while in practice it is possible for some clients to not have some of the tasks. Depending on the order of tasks, the system will have different trade-offs in terms of client drift and catastrophic forgetting. Intuitively, since the data from all the tasks are repeated across all clients over several rounds of training, the case of FMTL will have the best performance in terms of both client drift and catastrophic forgetting given a model with sufficient capacity. In the case of synchronous FCL, the problems of catastrophic forgetting and client drift are akin to that of standard continual learning and federated learning respectively. On the other hand, in asynchronous FCL the later rounds can overwrite
the knowledge accumulated by the clients in the former rounds aggravating the problem of client drift.

In this paper, we focus on the problem of additional client drift in asynchronous FCL. We use a dynamic architecture approach based on NetTailor [32] which has independent modules for each task thereby eliminating the problem of catastrophic forgetting and isolating the effect of client drift. We show that using adaptive federated optimization [38] can tackle this problem by showing its effectiveness by visualizing the client drift between models from consecutive rounds. We provide a comprehensive analysis of our system spanning different hyperparameters such as the number of communication rounds and learning rates. Further, we evaluate the performance with respect to challenges in a large-scale federated system such as scalability, data distribution, and stragglers. Our contributions are summarized as follows:

– We examine the problem of federated continual learning from the perspective of relative task order in the clients and identify client drift as the central challenge. We develop a framework for FCL based on the NetTailor approach to show the extent of client drift.
– We show that adaptive federated optimization (FedOpt) reduces the effect of client drift. With FEDADAM, we achieve an improvement of 2.45% in terms of average accuracy while reducing the client drift by 4.92% over standard FedAvg for a 5 client system trained on CIFAR100 split into 10 tasks.
– We provide a comprehensive view of the hyperparameter space by performing ablations studies on client and server learning rates, number of communication rounds, and local epochs. We analyze the robustness of the system to different challenges of federated learning such as non-IID distribution, stragglers, and scalability.

2 Related work

The popular continual learning methods can be broadly classified into three categories — (1) Regularization approaches (2) Dynamic Architecture approaches (3) Memory Replay and Complementary Learning System approaches [33,6]. Regularization-based approaches train a single model by using some form of regularization while learning new tasks so as to reduce the overwriting of the model parameters learned from the previous tasks [20,15,28,19,55]. Dynamic architecture approaches modify the network architecture according to the task either by adding additional modules or by using a subset of modules [11,13,18,24,50,57,49,9,34,35,32]. Memory replay methods store a subset of the data from the previous tasks or synthesize using a generative model and use it to retrain along with the new tasks and avoid the drop in performance on previous tasks [10,37,27,39,24,12]. Complementary learning systems use different modules to learn and store generic long-term knowledge and task-specific short-term knowledge [30,46,11].

While regularization approaches are generally more efficient as they use a single model for all the tasks, training such methods across multiple clients in a federated learning paradigm involves significant interference both between tasks
and data distribution in clients. On the other hand, since memory replay methods require storing a subset of past data to train the model on new tasks, it is not viable to retain data privacy while training such systems in a federated paradigm. Although dynamic architecture approaches require additional memory as we add more modules to the model architecture for each new task, they offer flexibility in terms of adding more tasks to each client independently. Moreover, the dynamic architecture approach decouples the orthogonal problems of catastrophic forgetting and client drift resulting in a simpler design. NetTailor [32] is one of the state-of-the-art continual learning methods based on the dynamic architecture approach which uses a common backbone network and adds additional lightweight task-specific branches resulting in a simple framework to implement continual learning without catastrophic forgetting. This provides a natural way of aggregating only the task-specific parameters to share knowledge between clients and hence is easily scalable to a large number of tasks. This further reduces the communication cost since only the gradients of task specific lightweight parameters are shared. We use NetTailor [32] as the candidate approach to show our findings. However, our framework can be extended to other continual learning methods as well.

Vanilla federated averaging (FedAvg) [30] suffers a performance drop when the data is not distributed in an independent and identical (non-IID) fashion among the clients [56]. There have been multiple works to improve federated averaging such as FedProx [23], which uses a regularization term to stabilize federated training, and FedMA [48], which performs a layer-wise matching and averaging of the models to handle heterogeneous data. The authors of SCAFFOLD [17] use control variates to counter the effect of client drift and stabilize the training process. Further, the authors of Adaptive Federated Optimization (FedOpt) [38] generalize the FedAvg and apply adaptive optimization algorithms such as ADAM and ADAGRAD in the federated aggregation process. This adaptive optimization makes the federated aggregation more smooth as a result provides an effective way to avoid client drift in federated continual learning systems.

To the best of our knowledge, FedWeIT [51] is the only work that attempts to address the problem of federated continual learning. FedWeIT uses weighted inter-client knowledge transfer to share constructive knowledge between clients having related tasks and avoid interference between clients having unrelated tasks. It achieves this by decomposing the network weights into global federated parameters and sparse task-specific parameters and the knowledge sharing is made selective by sharing a weighted combination of their task-specific parameters. We take an alternative approach of dynamic architecture with disjoint task-specific parameters thereby removing task interference altogether and at the same time reducing the overall communication cost. While FedWeIT provides a more efficient model with shared weights, our framework achieves higher accuracy since the interference between different task parameters is eliminated as well as effect of client drift is reduced with FEDADAM. We provide a comparison with FedWeIT on CIFAR100 dataset in section 5.3. Further, we provide experiments on the Decathlon benchmark which is a well-known benchmark for
evaluating continual learning algorithms as compared to the simulated Non-IID 50 dataset used in FedWeIT. Additionally, we provide an in-depth analysis of different hyperparameters and challenges of federated learning such as scalability, non-IID distribution, robustness to stragglers.

3 Preliminaries
3.1 Federated Learning
A standard federated learning system consists of a set of $N$ clients and a central server. The central server broadcasts the initial model parameters (say $M^{(0)}(s)$) to the clients to initialize the training process. Each of the clients $c = 1, 2, ..., N$, store their own private data $D^{(c)}$ which is used to obtain locally trained model (say $M^{(c)}$). The clients accumulated gradients from local training and periodically communicate them to the central server for aggregation. Each iteration of communication of the gradients from clients to the server followed by broadcasting of the updated model back to the clients is defined as a federated learning round. At round $r$, the gradients $\Delta M^{(r)}(c)$ from a subset of participating clients $S^r(\ll N)$ is sent to the central server. The central server performs a weighted average of the gradients to obtain the updated global model as shown in Eq. 1.

$$M^{(r+1)}(s) = M^{(r)}(s) + \frac{1}{\sum_{c \in S^r} |D^{(r)}(c)|} \sum_{c \in S^r} |D^{(r)}(c)| \Delta M^{(r)}(c),$$ (1)

where $|D^{(c)}(r)|$ denotes the number of data samples used for local training in client $c$ at round $r$. This aggregation of the model updates from the clients is described as FedAvg algorithm in [30].

3.2 Client Drift
The major problem with federated learning systems is that of client drift [17] where the updates from one client overwrites the model weights learned with data from other clients. This can be estimated by observing the distance from the initial model and the trained model at a client. Formally, client drift at round $r$ is defined as the expected amount by which the client model deviates from the starting model received from the server,

$$\mathcal{E}^{(r)} = \frac{1}{KN} \sum_{k=1}^{K} \sum_{c=1}^{N} \mathbb{E}[[|M^{(r)}_{(c,k)} - M^{(r-1)}_{s}|^2]],$$ (2)

Here, $c \in [N]$ denotes the clients, and $k \in [K]$ denotes the local training epochs over the client’s dataset. $|| \cdot ||$ is a distance metric such as Euclidean or Cosine distance defined over the model weights.

3.3 NetTailor
The architecture of NetTailor [32] involves a common backbone network trained on a large dataset such as ImageNet and the task-specific blocks interspersed
between the layers. Formally, a standard convolutional network implements the following function

\[ f(x) = (G_L \circ G_{L-1} \circ \cdots \circ G_1)(x), \]  

(3)

where \( G_l \) is a function at layer \( l \) such as convolution, spatial pooling, normalization, etc. NetTailor uses this as a backbone and augments it with a set of task-specific proxy layers \( \{P_t^{(l',l)}(\cdot)\}_{l'=1}^{l-1} \) that introduce skip connections between layer \( l' \) and layer \( l \) for each task \( t \). These proxy branches are computationally lightweight consisting of max pooling and 1x1 convolution layers. For example, compared to standard ResNet blocks which contain two 3x3 convolution layers, these contain \( \frac{1}{18} \) of parameters and perform \( \frac{1}{18} \) of operations. In practice, it is often redundant to have proxy layers connecting all the previous layers. Hence, the number of proxy layers can be limited to the previous \( k \) layers. The activation of each layer is computed by taking a linear combination of the backbone layer and the proxy layers. Formally, the activations at layer \( l \) for task \( t \) can be expressed as,

\[ x_l = \alpha_l^l G_l(x_{l-1}) + \sum_{l'=\max(l-k,1)}^{l-1} \alpha_{l'}^l P_t^{(l',l)}(x_{l'}). \]  

(4)

Here \( \{\alpha_l^l\}_{l'=\max(l-k,1)}^{l-1} \in [0,1] \) are the set of parameters denoting the importance of the branches that are used to enable or disable the branches. Since all the branches are not necessary for all tasks, this architecture is further pruned according to the complexity of tasks. The fact that NetTailor keeps the backbone network frozen and only updates the proxy branches aids in reducing the communication cost in federated learning since we only have to communicate the task-specific parameters and their gradients.

4 Federated Continual Learning Framework

The objective of our work is to showcase the challenge of client drift in federated continual learning systems and show that adaptive federated optimizers can address the effect of client drift. The major challenge in a federated continual learning system is the relative order of tasks in the clients. As illustrated in Fig. 1b, we define three different ordering of tasks namely,

- **Case 1:** Federated Multi-Task Learning: Shuffled task order with repetition.
- **Case 2:** Synchronous FCL: Common task order.
- **Case 3:** Asynchronous FCL: Shuffled task order without repetition.

To study the effect of client drift in the above settings, we design a structured FCL framework based on NetTailor. The model in each client comprises of backbone parameters \( G \) and task-specific parameters \( P \). As shown in Fig. 2 each client contains a frozen backbone network and a set of trainable task-specific blocks. We use \( P_{(t,c)} \) to denote the set of task-specific proxy layers of task \( t \) in
Fig. 2: Illustration representing our overall approach. The model in each client comprises a backbone network and a set of task-specific proxy layers for each task. The backbone network is pretrained using a large dataset like ImageNet hence contains the generic knowledge and is kept frozen. Each client independently trains only the task-specific blocks using the local data available. In every communication round, the gradients of the proxy layers are communicated to the server which aggregates these proxy layers and broadcasts them back to all the clients.

Client \( c \) at round \( r \), and \( P_{(t,s)}^r \) to represent the set of corresponding proxy layers of the server. At round \( r \), each client \( c \) has the global parameters sent from the server \( (P_{(t,c,0)}^r = P_{(t,s)}^{r-1}) \). We then train the local proxy layers in each client for \( K \) local epochs on the local client data as

\[
P_{(t,c,k)}^r = P_{(t,c,k-1)}^r - \mu_{(t,c,k)}^r \nabla P_{t,c,k-1}^r,
\]

where \( \mu_{(t,c,k)}^r \) is the local learning rate at \( k \)-th local epoch on client \( c \) on task \( t \) and \( \nabla P_{t,c,k-1}^r \) is the proxy layer’s gradient from the previous local epoch \( k - 1 \).

Once the clients have performed local training, we obtain the model updates \( \Delta P_{(t,c)}^r \), i.e. the difference in the initial model received from the server and the final model after training on the local data.

\[
\Delta P_{(t,c)}^r = P_{(t,c,K)}^r - P_{(t,s)}^{r-1}.
\]

The server collects all the updates from the clients and performs federated aggregation. In the case of standard FedAvg, the server updates the model with the average of the gradients.

\[
P_{(t,s)}^r = P_{(t,s)}^{r-1} + \sum_{c \in S_t} (\Delta P_{(t,c)}^r).
\]

This is equivalent to applying SGD with pseudo-gradient \( \sum_{c \in S_t} (\Delta P_{(t,c)}^r) \) with a learning rate \( \eta = 1 \). In a more general form, the server update can be written as:

\[
P_{(t,s)}^r = P_{(t,s)}^{r-1} + \eta_t^r \Delta_t^r.
\]
Table 1: Calculation of second order momentum of pseudo-gradients in different adaptive federated optimizers [38].

| Method       | Update Rule for Second Moment |
|--------------|-------------------------------|
| FEDADAGRAD   | $v_r^t = v_r^{t-1} + \Delta r^2$ |
| FEDYOGI      | $v_r^t = v_r^{t-1} - (1 - \beta_2)\Delta r^2 \text{sign}(v_r^{t-1} - \Delta r^2)$ |
| FEDADAM      | $v_r^t = \beta_2 v_r^{t-1} + (1 - \beta_2)\Delta r^2$ |

Algorithm 1 FCL Training Method. [Case 1 Case 2 Case 3]

**Input:** Clients $c \in [N]$ Tasks $t \in [T]$, No. of rounds $R$, Local datasets $D_c \forall c \in [N]$

**Output:** Trained task-specific blocks $P_{(t,s)} \forall t \in [T]$

1: for client $c = 1, \ldots N$ do
2: \hspace{1cm} task$_{order,c} = \text{permute}(1\ldots T)$
3: end for
4: for round $r = 1, \ldots R$ do
5: \hspace{1cm} $Q = R/T$
6: \hspace{1cm} for client $c = 1, \ldots N$ do
7: \hspace{2cm} task $t \sim [T]$ sampled uniformly at random
8: \hspace{2cm} task $t = \lceil r/Q \rceil$
9: \hspace{2cm} task $t = \text{task}_{order,c}(\lceil r/Q \rceil)$
10: \hspace{2cm} Local update for $K$ epochs on $P_{r,c}^{(t)}$ (Eq. )
11: \hspace{2cm} $S_t^r = S_t^r \cup c$
12: end for
13: for task $t = 1, \ldots T$ do
14: \hspace{1cm} if $S_t^r \neq \phi$ then
15: \hspace{2cm} Global update on task $t$, $P_{r,s}^t$
16: \hspace{2cm} end if
17: end for
18: end for

where $\eta_r^{(t)}$ is the server learning rate at round $r$ for task $t$, and $\Delta r^t$ is the pseudo-gradient which is $\sum_{c \in S_t^r} (\Delta P_{r,c}^{(t)})$ in case of FEDSGD. With the notion of momentum with a decay parameter $\beta_1$, the pseudo gradient can be expressed as $\Delta r^t = \beta_1 \Delta r^{t-1} + (1 - \beta_1) \sum_{c \in S_t^r} (\Delta P_{r,c}^{(t)})$. The federated aggregation can be further generalized to more adaptive gradient descent algorithms. This family of algorithms is described as FedOpt in [38]. The update rule in FedOpt is expressed as,

$$P_{r,s}^{(t)} = P_{r-1,s}^{(t)} + \eta_r^{(t)} \frac{\Delta r^t}{\sqrt{v_r^t + \tau}}$$

where $v_r^t$ is the exponential average of the square of the pseudo-gradient (second moment) that captures the variance. This is calculated differently in each of the variants of FedOpt as summarized in Table 1. The parameter $\tau$ controls the degree of adaptiveness of the algorithm with smaller $\tau$ implying more adaptive. FedOpt keeps track of the history of gradients over several rounds and updates the model in a smooth fashion, it results in avoiding abrupt changes to the global model thereby reducing client drift.
We define a phase \((Q)\) as the number of rounds a client works on each task and break the overall training into \(T\) phases. In both cases 2 and 3, we change the task at each client only after \(Q\) rounds. For instance, in the example shown in Fig. 1b, the total rounds \((R)\) is 16 and the number of tasks \((T)\) is 4 resulting in \(Q = 4\). Hence, the task is changed in all the clients after every 4 rounds. In cases 2 and 3, we divide \(R\) rounds of federated training into \(T\) phases of \(Q\) rounds each. Then, we perform federated aggregation on the updates from all the clients with one task every \(Q\) rounds. This ensures that each client performs an equal amount of training in terms of number of local epochs for all three cases resulting in a fair comparison. Algorithm 1 provides a flow of the FCL for different task ordering. We focus mainly on the case of shuffled task order without repetition depicting asynchronous FCL (Case 3) to perform extensive analysis and use cases 1 and 2 for comparison.

5 Experiments

We conduct extensive experiments spanning different aspects of federated learning as well as continual learning on three benchmark datasets — CIFAR100 \cite{krizhevsky2009learning}, MiniImagenet \cite{vinyals2016matching} and Decathlon Challenge for Continual Learning \cite{rebuffi2018icarl}. CIFAR100 is a standard benchmark dataset widely used in computer vision containing 50000 training images and 10000 test images with 32x32 pixel dimensions and 3 channels. We split the training data into 10 tasks of 10 classes each. MiniImagenet is a 100 class dataset sampled from the ImageNet data. The resolution of the images is reduced to 84x84. Similar to CIFAR100, we divide the miniImagenet into 10 tasks of 10 classes each. Decathlon benchmark is a collection of ten well-known datasets used to benchmark continual learning methods on visual domain. Each dataset in the collection is defined as a separate task. Across all three datasets, the test set is held out and all the reported evaluation is performed on the entire test set. We divide the datasets across the clients uniformly at random to obtain identically distributed data (IID) and use Dirichlet distribution to simulate non-IID scenarios. We use the IID case across our experiments to identify the effect of varying each parameter independently and study the impact of non-IID distribution in section 5.4. We use standard ResNet18 \cite{he2016deep} as the backbone model across all our experiments. We examine the behavior of client drift in different ordering of tasks and show the extent of the problem in 5.1. While we use all three datasets mentioned above for showing the effectiveness of our framework in section 5.2, we use CIFAR100 as the candidate dataset for comparing with related work in section 5.3. We use the set of hyperparameters shown in Table 2 as default values and perform a principled ablation study by varying each of the parameters in the section 5.4.

**Evaluation Metrics:** The key performance metric is the average accuracy of the model on all tasks. Additionally, we also keep track of a metric to measure the client drift. Similar to backward transfer in continual learning \cite{lopez2017gradient}, which measures the effect of learning a new task \(t\) on the performance of previous tasks \(t' < t\), we define federated negative backward transfer \(\text{BWT}_f\) to quantify the
Table 2: Default hyperparameters used across all the experiments unless specified otherwise.

| Parameter               | Value                  | Parameter               | Value                  |
|-------------------------|------------------------|-------------------------|------------------------|
| Dataset                 | CIFAR100               | Client Optimizer        | SGD                    |
| Backbone Model          | ResNet18               | Client LR (µ)           | 0.05                   |
| No. of Tasks (T)        | 10                     | No. of Local Epochs (K) | 2                      |
| No. of Clients (N)      | 5                      | Data Distribution       | IID                    |
| No. of Rounds (R)       | 300                    | Server Optimizer        | FEDADAM                |
| No. of Rounds per phase | Q=R/T=30               | Server LR (η)           | 0.5                    |

(a) The model accuracy on each of the tasks in different order of tasks. 
(b) Client drift in each of the three order of tasks.

Fig. 3: Illustration of client drift in asynchronous FCL. (a) Comparison of the different ordering of tasks. Note the performance drop in asynchronous FCL (Case 3) compared to synchronous FCL (Case 2) in most of the tasks, while FMTL performs the best among the three cases. (b) Visualization of client drift in terms of cosine distances between the models from consecutive rounds for the three cases of the relative order of tasks.

Client drift introduced by the latter clients on the learned model. We capture the test accuracy of all tasks at each federated round resulting in a accuracy matrix $A \in \mathbb{R}^{T \times R}$ and define the evaluation metrics of average accuracy ($ACC = \frac{1}{T} \sum_{t=1}^{T} A_{t,R}$) and $BWT_f$ as follows:

$$BWT_f = \frac{1}{T^2} \sum_{t=1}^{T} \sum_{p=1}^{T} (A_{t,p,Q} - A_{t,p,Q-1}).$$

Here, Q is the number of rounds per phase. With $BWT_f$, we are capturing the drop in accuracy resulting from new clients updating the model after every phase and take the average over all the tasks.

5.1 Client Drift in Different Order of Tasks
Here we compare the different order of tasks described in section 4. We present the task-wise accuracy for all three cases on the CIFAR100 dataset in Fig. 3a. Since the updates from different clients is repeated in FMTL, the overwriting is avoided resulting in the highest performance among the three cases (i.e. baseline average accuracy of 82.79%). In synchronous FCL, since all the clients are synchronously training on the same task, the client drift is at a minimum hence the
Addressing Client Drift in FCL

Table 3: Summary of performance of different variants of adaptive optimizers on three different datasets. We highlight the top two optimizers achieving high accuracy and low $BWT_f$ for each dataset. Adaptive optimizers consistently perform well both in achieving high accuracy and reducing the $BWT_f$.

| Dataset            | Optimizer   | $\eta = 1.0$ | $\eta = 0.5$ | $\eta = 0.1$ |
|--------------------|-------------|--------------|--------------|--------------|
| CIFAR100           | FEDSGD      | 77.59        | 79.27        | 78.51        | -5.76       | -1.04       | -0.63       |
|                    | FEDADAM     | 77.85        | **80.04**    | 77.98        | -3.61       | -0.84       | -0.55       |
|                    | FEDADAGRAD  | 77.55        | 78.33        | 78.98        | -4.42       | -1.39       | -0.51       |
|                    | FEDYOGI     | 76.81        | 79.33        | 78.97        | -4.44       | -1.77       | **-0.50**   |
| MiniImageNet       | FEDSGD      | 56.01        | 56.82        | 52.44        | -1.86       | -0.45       | -0.12       |
|                    | FEDADAM     | 56.89        | 57.28        | 53.73        | -1.75       | -0.16       | **-0.08**   |
|                    | FEDADAGRAD  | 56.81        | 57.14        | 52.23        | -1.72       | -0.23       | -0.08       |
|                    | FEDYOGI     | 56.75        | **57.34**    | 52.98        | -1.98       | -0.21       | -0.11       |
| Decathlon Challenge| FEDSGD      | 73.44        | 74.62        | 74.01        | -3.45       | -1.42       | -0.84       |
|                    | FEDADAM     | 73.24        | 74.74        | 74.52        | -2.82       | -1.04       | -0.67       |
|                    | FEDADAGRAD  | 73.82        | **75.03**    | 74.61        | -3.23       | -1.45       | -0.53       |
|                    | FEDYOGI     | 72.56        | 74.38        | 73.63        | -3.16       | -1.00       | **-0.46**   |

accuracy is high. However, in the case of asynchronous FCL, where the tasks are shuffled and there is no repetition, updates from the later rounds can overwrite the information learned from the clients in earlier rounds resulting in a significant performance drop. We use cosine distance between the models following eq. 2 to quantify and visualize client drift. We take the weights of the fully connected layers for each task as the proxy for the model and calculate the cosine distance between the models from consecutive rounds and plot for all three cases in Fig. 3b. Note the spikes in cosine distance in the case of asynchronous FCL indicates that the clients in the later rounds overwrite the information learned from the earlier rounds highlighting the existence of the problem of aforementioned client drift in federated continual learning systems. In contrast, the case of FMTL shows significant deviation only during the initial stages of learning and becomes stable once the model is sufficiently trained. We show that using adaptive optimizers can reduce the impact of client drift and present our findings in the next section.

5.2 Adaptive Optimizers and Client Drift

In this section, we highlight the advantage of using adaptive optimizers in federated aggregation to avoid the aforementioned problem of client drift. We compare the variants of adaptive optimizers with three different learning rates of 1.0, 0.5, and 0.1 on three datasets — CIFAR100, MiniImageNet, and Decathlon Challenge (shown in Table 3). In general, we observe a trend of adaptive optimizers reducing the client drift as reflected with low negative $BWT_f$. At the same time, the final accuracy reached is higher than the baseline model that uses FedAvg (i.e., FEDSGD with a learning rate of 1.0). For example, FEDSGD with a learning rate of 1.0, which is equivalent to standard FedAvg, suffers from a severe client drift resulting in a 5.76% drop in accuracy for CIFAR100. In contrast, FEDADAM reduces the $BWT_f$ to 3.61% for the same setting.
Fig. 4: Visualization of training progress of FEDADAM as compared to FEDSGD on one of the tasks of CIFAR100. (a) Shows the trend of model accuracy on one of the tasks over the course of training demonstrating the effectiveness of the FEDADAM optimizer in minimizing the sudden accuracy drops seen in FEDSGD. (b) The steep spikes in client drift measured in terms of cosine distances between the models from consecutive rounds illustrate the advantage of using FEDADAM over FEDSGD.

To further understand the impact of client drift and the effectiveness of using an adaptive optimizer we zoom in on the training progress of Task 1 on the CIFAR100 dataset across federated rounds in Fig. 4a. Note the sharp drops in accuracy at the boundaries of phases (at rounds 180 and 210) with FEDSGD which indicate that the updates from the current rounds are overwriting the model learned using the data from clients in the previous rounds. We compare the cosine distances for FEDADAM with learning rates 0.5 and 0.1 to the baseline FEDSGD with a learning rate of 1.0. FEDADAM avoids sudden jumps in the model updates by using the history of gradients to make the updates smooth. At the same time, if we use a lower learning rate, the updates themselves become small, hence, the convergence rate of the model is adversely affected. We expand on this further by studying the behavior of the training method for different learning rates and the number of rounds in section 5.4.

5.3 Comparison to Previous Work

We compare our asynchronous FCL results with FEDADAM training against previous work of FedWeIT [51] on CIFAR100 dataset in Table 4. Note that while the dataset partition is different since the reported accuracy is the average across all tasks for CIFAR100, the comparison is fair. Our method outperforms FedWeIT by a significant margin in terms of accuracy while having <1% client drift. The improved performance can be attributed to the separate task parameters of the NetTailor architecture thereby eliminating the catastrophic forgetting due to interference between the tasks. Also, the FEDADAM optimization helps in reducing the overall client drift. Having separate parameters for each task aids in scaling our framework to more tasks as we can add an unlimited number of tasks without interfering with each other whereas it is uncertain how FedWeIT will scale to additional tasks. Additionally, we compare the model size and average communication cost per round for sharing the gradients from clients to server.
Table 4: Comparison to previous work in terms of accuracy, forgetting, model size and communication cost on CIFAR100 dataset. C2S and S2C denote communication cost from client to server and server to client respectively.

| Method       | Accuracy | $BWT_f$ | Model Size | Comm. cost per round (C2S + S2C) |
|--------------|----------|---------|------------|----------------------------------|
| FedWeIT [51] | 55.16%   | -       | 75MB       | 18.5MB + 53.5MB                  |
| Ours         | 80.04%   | 0.84%   | 90MB*      | 4.7MB + 4.7MB                    |

* 43MB (backbone) + 10 tasks × 4.7MB (per task)

Fig. 5: Ablation study of different hyperparameters involved. We report the performance of the system in terms of ACC and $BWT_f$. The heatmaps are color-coded with green for higher accuracy (higher is better) and red for larger client drift (lower magnitude is better). (a) The trend of Accuracy and $BWT_f$ with local and global learning rates. (b) Trend of accuracy and $BWT_f$ with the number of rounds and local epochs. (c) Performance trend with varying the number of clients. (d) Non-IID Trend. (e) Performance trend in the presence of stragglers. (C2S) and broadcasting the updated model to the clients (S2C) for a 5 client system trained on 10 tasks. While our model size is higher compared to FedWeIT, the communication cost per round is significantly lower since we are only communicating the gradients of the task-specific modules. This can be further reduced by applying gradient compression methods [2611].

5.4 Ablation Studies

Number of rounds and local epochs: The number of local epochs and the total number of federated rounds are important factors in achieving convergence in federated learning systems. Here, we vary the number of local epochs from 1 to 5 for a different number of federated rounds ranging from 100 to 400. As summarized in Fig. 5a, we observe that having a large number of federated rounds reduces the accuracy as client drift accumulates over several rounds.

Server and client learning rates: Here we vary both client and server learning rates and measure the final performance of the system (see Fig. 5b). The learning rate at the client dictates how effectively the local models are being learned. This follows the trend of standard deep learning where using a lower learning rate will make the convergence slower but increasing the learning rate after some point makes the training unstable resulting in a diverging model. Therefore, we see an improvement in average accuracy as we increase the client learning rate up to 0.05 but the accuracy drops as we go to 0.1. Similarly, decreasing the
server learning rate makes the federated aggregation process smooth, resulting in improved performance both in terms of accuracy and client drift. However, reducing it further makes the updates extremely small resulting in slower convergence. Hence, for a fixed number of rounds, the final model has lower accuracy. **Scalability:** In this experiment (Fig. 5c), we vary the number of clients to observe the final accuracy reached by the models. Naturally, as the number of clients increases, there are more opportunities for newer clients to overwrite the learned model. Hence the performance of the system is reduced as the number of clients increases. **Non-IID data:** Since practical federated learning systems often have a non-IID data distribution among the clients, we simulate the skewed distribution (Fig. 5d) by sampling the proportions of each class based on the Dirichlet distribution following previous works [48,54]. By varying the parameter (alpha) of the Dirichlet distribution, we can control the skewness of the data distribution. Here, having a low alpha indicates that the data distribution is more skewed. As we vary the alpha value progressively from 8 to 0.5, we observe a decrease in performance both in terms of accuracy as well as $BWT_f$. This implies that the skewness in the system amplifies the client drift. **Stragglers:** Finally, it is common to have heterogeneous clients with different capabilities in terms of compute, memory and communication bandwidth. Hence, it is not always feasible to assume all the clients communicate their gradients in time to participate in the federated aggregation process. We measure the robustness of the system with respect to stragglers by varying the probability of the clients failing to communicate their gradients to the server. We progressively increase the probability of each client showing a straggler behavior (Fig. 5e). We observe that while the performance goes down it is still within less than a 2% drop from the baseline case, where, all the clients are guaranteed to communicate the updates. This shows the robustness of the system to stragglers.

6 Conclusion and Limitations

In this work, we present a framework for federated continual learning and identify client drift as the central challenge. We show that using adaptive federated optimizers can stabilize the federated aggregation and hence reduce the client drift. We provide a comprehensive analysis of different hyperparameters impacting the performance of the system.

**Limitations:** While we propose a framework to perform federated continual learning with reduced client drift, there are numerous unsolved challenges for achieving a scalable and robust federated continual learning system. For instance, while we reduce client drift with adaptive optimizers, the problem is not fully solved. We observe in our ablation studies that as we increase the number of rounds, the clients in the later rounds eventually overwrite the model thereby reducing the accuracy of the final model. Additionally, while NetTailor based approach removes catastrophic forgetting, the model size scales linearly with number of tasks making it impractical when we have large number of tasks. It is
necessary to examine other continual learning methods that use a single model across all tasks such as Gradient Projection Memory (GPM) [42] and Relevance Mapping Networks (RMN) [18]. This will involve two orthogonal challenges of avoiding overwriting the model between tasks as well as between clients and additional scalability challenges which we leave for future work.
References

1. Albasyoni, A., Safaryan, M., Condat, L., Richtárik, P.: Optimal gradient compression for distributed and federated learning. arXiv preprint arXiv:2010.03246 (2020)
2. Azizpour, H., Sullivan, J., Carlsson, S., et al.: Cnn features off-the-shelf: An astounding baseline for recognition. In: CVPRW, pp. 512–519 (2014)
3. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018)
4. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: Continual learning with tiny episodic memories (2019)
5. Chen, Z., Liu, B.: Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 12(3), 1–207 (2018)
6. Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
7. Dinh, C.T., Vu, T.T., Tran, N.H., Dao, M.N., Zhang, H.: Fedu: A unified framework for federated multi-task learning with laplacian regularization. arXiv preprint arXiv:2102.07148 (2021)
8. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International conference on machine learning. pp. 647–655. PMLR (2014)
9. Draelos, T.J., Miner, N.E., Lamb, C.C., Cox, J.A., Vineyard, C.M., Carlson, K.D., Severa, W.M., James, C.D., Aimone, J.B.: Neurogenesis deep learning: Extending deep networks to accommodate new classes. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 526–533. IEEE (2017)
10. French, R.M.: Pseudo-recurrent connectionist networks: An approach to the ‘sensitivity-stability’ dilemma. Connection Science 9(4), 353–380 (1997)
11. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211 (2013)
12. Guo, Y., Liu, M., Yang, T., Rosing, T.: Improved schemes for episodic memory-based lifelong learning. arXiv preprint arXiv:1909.11763 (2019)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Hinton, G.E., Plaut, D.C.: Using fast weights to deblur old memories. In: Proceedings of the 9th Annual Conference of the Cognitive Science Society. pp. 177–186 (1987)
15. Jung, H., Ju, J., Jung, M., Kim, J.: Less-forgetting learning in deep neural networks. arXiv preprint arXiv:1607.00122 (2016)
16. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
17. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T.: Scaffold: Stochastic controlled averaging for on-device federated learning. CoRR abs/1910.06378 (2019), http://arxiv.org/abs/1910.06378
18. Kaushik, P., Gain, A., Kortylewski, A., Yuille, A.: Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. arXiv preprint arXiv:2102.11343 (2021)
19. 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 114(13), 3521–3526 (2017)
20. Konečný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
22. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine 37(3), 50–60 (2020)
23. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)
24. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: International Conference on Machine Learning. pp. 3925–3934. PMLR (2019)
25. Li, T., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence 40(12), 2935–2947 (2017)
26. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv preprint arXiv:1712.01887 (2017)
27. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. Advances in neural information processing systems 30, 6467–6476 (2017)
28. Maltoni, D., Lomonaco, V.: Continuous learning in single-incremental-task scenarios. Neural Networks 116, 56–73 (2019)
29. Marfoq, O., Neglia, G., Bellet, A., Kameni, L., Vidal, R.: Federated multi-task learning under a mixture of distributions. Advances in Neural Information Processing Systems 34 (2021)
30. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) (2017), http://arxiv.org/abs/1602.05629
31. Mohri, M., Sivek, G., Suresh, A.T.: Agnostic federated learning. In: International Conference on Machine Learning. pp. 4615–4625. PMLR (2019)
32. Morgado, P., Vasconcelos, N.: Nettailor: Tuning the architecture, not just the weights. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3044–3054 (2019)
33. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks 113, 54–71 (2019)
34. Part, J.L., Lemon, O.: Incremental on-line learning of object classes using a combination of self-organizing incremental neural networks and deep convolutional neural networks. In: Workshop on Bio-inspired Social Robot Learning in Home Scenarios (IROS), Daejeon, Korea (2016)
35. Part, J.L., Lemon, O.: Incremental online learning of objects for robots operating in real environments. In: 2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob). pp. 304–310. IEEE (2017)
36. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. arXiv preprint arXiv:1705.08045 (2017)
37. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
38. Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, H.B.: Adaptive federated optimization. arXiv preprint arXiv:2003.00295 (2020)
39. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv preprint arXiv:1810.11910 (2018)
40. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science 7(2), 123–146 (1995)
41. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
42. Saha, G., Garg, I., Roy, K.: Gradient projection memory for continual learning. arXiv preprint arXiv:2103.09762 (2021)
43. Sarwar, S.S., Ankit, A., Roy, K.: Incremental learning in deep convolutional neural networks with partial network sharing. IEEE Access 8, 4615–4628 (2019)
44. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. arXiv preprint arXiv:1705.08690 (2017)
45. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. Advances in neural information processing systems 30 (2017)
46. Soltoggio, A.: Short-term plasticity as cause–effect hypothesis testing in distal reward learning. Biological cybernetics 109(1), 75–94 (2015)
47. Sun, Q., Liu, Y., Chua, T., Schiele, B.: Meta-transfer learning for few-shot learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
48. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging. arXiv preprint arXiv:2002.06440 (2020)
49. Xiao, T., Zhang, J., Yang, K., Peng, Y., Zhang, Z.: Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In: Proceedings of the 22nd ACM international conference on Multimedia. pp. 177–186 (2014)
50. Xu, J., Zhu, Z.: Reinforced continual learning. arXiv preprint arXiv:1805.12369 (2018)
51. Yoon, J., Jeong, W., Lee, G., Yang, E., Hwang, S.J.: Federated continual learning with weighted inter-client transfer. In: International Conference on Machine Learning. pp. 12073–12086. PMLR (2021)
52. Yoon, J., Kim, S., Yang, E., Hwang, S.J.: Scalable and order-robust continual learning with additive parameter decomposition. arXiv preprint arXiv:1902.09432 (2019)
53. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
54. Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian nonparametric federated learning of neural networks. In: International Conference on Machine Learning. pp. 7252–7261. PMLR (2019)
55. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017)
56. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
57. Zhou, G., Sohn, K., Lee, H.: Online incremental feature learning with denoising autoencoders. In: Artificial intelligence and statistics. pp. 1453–1461. PMLR (2012)