Don't Memorize; Mimic The Past: Federated Class Incremental Learning Without Episodic Memory

Sara Babakniya [1] **Zalan Fabian** [2] **Chaoyang He** [3] **Mahdi Soltanolkotabi** [2] **Salman Avestimehr** [2]

Abstract

Deep learning models are prone to forgetting information learned in the past when trained on new data. This problem becomes even more pronounced in the context of federated learning (FL), where data is decentralized and subject to independent changes for each user. Continual Learning (CL) studies this so-called *catastrophic forgetting* phenomenon primarily in centralized settings, where the learner has direct access to the complete training dataset. However, applying CL techniques to FL is not straightforward due to privacy concerns and resource limitations. This paper presents a framework for federated class incremental learning that utilizes a generative model to synthesize samples from past distributions instead of storing part of past data. Then, clients can leverage the generative model to mitigate catastrophic forgetting locally. The generative model is trained on the server using data-free methods at the end of each task without requesting data from clients. Therefore, it reduces the risk of data leakage as opposed to training it on the client's private data. We demonstrate significant improvements for the CIFAR-100 dataset compared to existing baselines.

1. Introduction

Federated learning (FL) (McMahan et al., 2017; Konečnỳ et al., 2016) is a decentralized machine learning technique that enables privacy-preserving collaborative learning. In FL, multiple users (clients) train a common (global) model in coordination with a centralized node (server) without sharing personal data. In recent years, FL has attracted tremendous attention in both research and industry and has been successfully employed in various fields. Despite its popularity, deploying FL in practice requires addressing challenges such as resource limitation and statistical heterogeneity (Kairouz et al., 2021). Furthermore, there are still common assumptions in most FL frameworks that are far from reality. One such assumption is that the client's local data distribution does not change over time. However, in real-world (Shoham et al., 2019), users' data constantly evolve due to changes in the environment or trends. In such scenarios, the model must rapidly adapt to the incoming data while preserving performance in the past.

In the centralized setting, such problems have been explored in continual learning (Shin et al., 2017; Li & Hoiem, 2017). Despite all the significant progress for the centralized problems, most methods cannot be directly employed in the FL setting due to inherent differences between the two settings. For instance, experience replay (Rolnick et al., 2019) is a popular approach, where a portion of past data points is saved to maintain some representation of past distributions throughout the training. However, deploying experience replay in FL has resource and privacy limitations. It requires clients to store and keep their data which may not be possible because of privacy reasons. This can be highly important, especially in cases for example a service provider can store its customer's data for only a short time. Besides, even storing is possible; such data overhead increase the memory usage of already resource-limited clients.

To address the above problems, we propose MFCL, *Mimicking Federated Continual Learning*. In particular, MFCL is based on training a generative model in the server and sharing it with clients to sample synthetic examples of past data instead of clients storing their data. The generative model training is data-free in the sense that it only requires the global model without any form of training data from the clients. This is specifically important because this step does not require powerful clients and does not cause any extra data leakage from them. Finally, our experiments demonstrate improvement by 20% in average accuracy while reducing the training overhead of the clients.

We summarize our contributions below:

[1]Department of Computer Science, University of Southern California, Los Angeles, USA [2]Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, USA [3]FedML. Correspondence to: Sara Babakniya <babakniy@usc.edu>.

Workshop of Federated Learning and Analytics in Practice, colocated with 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA.

- We propose a novel framework to tackle the problem of federated class incremental learning more efficiently. Our framework specifically targets applications where past samples are unavailable.
- We point out potential issues with relying on client-side memory for FCL, and propose using a generative model trained by the server in a *data-free manner* to reduce catastrophic forgetting while preserving privacy.
- We demonstrate the efficacy of our method in more realistic scenarios with a larger number of clients and a more challenging dataset (CIFAR-100).

2. Related Work

Continual Learning. Catastrophic forgetting (McCloskey & Cohen, 1989) is a fundamental problem: when we train a model on new examples, its performance on past data degrades. This problem is investigated in continual learning (Zenke et al., 2017), and the goal is for the model to learn new information while preserving its old ones.

Recent works focus on three scenarios, namely task-, domain- and class-incremental learning (Van de Ven & Tolias, 2019). In *Task-IL*, tasks are disjoint, and the output spaces are separated by task IDs provided during training and test time. For *Domain-IL*, the output space is still the same, but the task IDs are no more provided. Finally, in *Class-IL*, new tasks introduce new classes to the output space, and the number of classes increases incrementally. Among these scenarios, we focus on **Class-IL**, which is the more challenging and realistic, especially in FL. In the FL applications, there is no task ID available, and it is preferred to learn a *single* model useable for all the observed data.

Federated Continual Learning. In Federated Continual Learning (FCL), the main focus is to adapt the global model to new data while maintaining knowledge of past data, all under the standard restrictions of FL. This important problem has only gained attention very recently, and (Yoon et al., 2021) is the first paper on this topic. It focuses on Task-IL, which requires a unique task id per task during inference. Furthermore, it adapts separate masks per task to improve personalized performance without preserving a common global model. This setting is considerably different than ours as we target class-IL with a single global model that can classify all the classes seen so far. (Ma et al.) employs knowledge distillation using a surrogate dataset. (Dong et al., 2022) relaxes the problem as clients have access to large memory to save the old examples and share their data which is different from the standard FL setting. (Jiang et al., 2021; Priyanshu et al., 2021; Usmanova et al., 2021) explore the FCL problem in domains other than image classification.

This work focuses on Class-IL for supervised image classification without memory replay, which has been also discussed in (Qi et al., 2023; Hendryx et al., 2021). However, (Hendryx et al., 2021) allows overlapping classes between tasks and focuses on few-shot learning, which is different from the standard class-IL. The most closely related work to ours is (Qi et al., 2023), where authors propose FedCIL. This work also benefits from methods based on generative replay to compensate for the absence of old data and overcome forgetting. In FedCIL, clients train the discriminator and generator locally. Then, the server takes a consolidation step after aggregating the updates. In this step, the server generates synthetic data using all the generative models trained by the clients to consolidate the global model and improve the performance. The main difference between this work and ours is that in our work, the generative model is trained by the server in a data-free manner which can reduce clients' computation and does not require their private data.

Data-Free Knowledge Distillation. Knowledge distillation (KD)(Hinton et al., 2015) is a popular method to transfer knowledge from a well-trained teacher model to a (usually) smaller student model using at least a small portion of training data. However, in cases that such data is unavailable (e,g, privacy concerns), a new line of work (Chen et al., 2019; Haroush et al., 2020) proposes *data-free knowledge distillation*. In such methods, a generative model is used as a training data substitute. This model generates synthetic data such that the teacher model predicts them as their assigned label. Data-free KD has been previously used in FL (Zhu et al., 2021) as a solution for data heterogeneity. However, to the best of our knowledge, this is the first work that adapted such a technique in the context of FCL.

3. Federated Class-IL with MFCL

In federated class-IL, a shared model is trained on T tasks. However, the distributed nature of FL makes it distinct from the centralized version. In FL, users may join, drop out or change their data independently. Also, required data or computation power for some centralized algorithms may not be available due to privacy and resource constraints.

To address the mentioned problems, we propose MFCL. This algorithm includes two essential parts: *first*, at the end of each task, the server trains a generative model with data-free knowledge distillation methods to learn the representation of the seen classes. *Second*, clients diminish catastrophic forgetting by generating synthetic images from the generative model. This way, clients do not require memories to store old data. Also, since the server trains the generative model training without additional information, this step does not introduce new privacy issues. Finally, MFCL can help mitigate the data heterogeneity problem, as clients can synthesize samples from classes they do not own. Here, we explain the two key parts of our algorithm: server-side (Fig. 1. left) and client-side (Fig. 1. right).

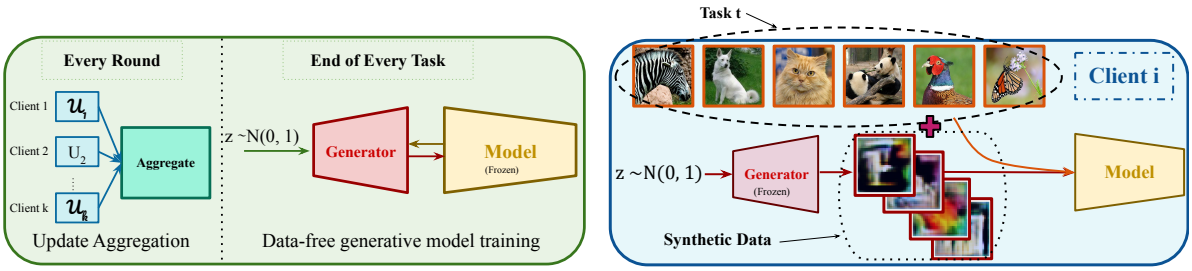

Figure 1. [Left] Aggregation (every round), generator training (end of each task). [Right] Clients train models using synthetic + local data.

3.1. Server-Side: Generative Model

The motivation for deploying a generative model is to synthesize images that mimic the old tasks and to avoid storing past data. However, training these generators on the client's side, where the training data exists, is *computationally expensive* and *requires a large amount of training data* and can be potentially *privacy concerning*. On the other hand, the server has only access to the global model and no data. We propose training a generative model on the server, but in a data-free manner, i.e., by means of model-inversion image synthesis (Yin et al., 2020; Smith et al., 2021). In such approaches, the goal is to synthesize images optimized with respect to the discriminator (global model). Then, the generative model is shared with the clients to be later used in sampling images during local training. To this aim, we utilize a generative model, $\mathcal{G}$, that takes noise $z \sim \mathcal{N}(0, \mathbf{I})$ as input and produces a synthetic sample $\tilde{x}$. In training this model, we employ the following training objectives.

Cross Entropy Loss. First, the synthetic data should be labeled correctly by the current discriminator model (global model or $\mathcal{F}$). Therefore, we employ cross entropy classification loss between its assigned label z and the prediction of $\mathcal{F}$ on synthetic data $\tilde{x}$. Note, that noise dimension can be arbitrary and greater than the current discovered classes of task t, and we only consider the first q dimension here, where $q = \sum_{i=1}^{t} |\mathcal{Y}^i|$. Then, we can define this loss as

$$\mathcal{L}_{CE} = CE(argmax(z[:q]), \mathcal{F}(\tilde{x})). \tag{1}$$

Diversity Loss. Synthesized images can suffer from a lack of class diversity, and we utilize information entropy (IE) (Chen et al., 2019) to solve this. For a probability vector $\mathrm{p} = (p_1, p_2, ..., p_q)$, IE is evaluated as $\mathcal{H}_{info}(\mathrm{p}) = -\frac{1}{q}\sum_i p_i \log(p_i)$. Therefore, diversity loss is defined as

$$\mathcal{L}_{div} = -\mathcal{H}_{info}(\frac{1}{bs}\sum_{i=1}^{bs} \mathcal{F}(\tilde{x}_i)). \tag{2}$$

This loss measures the IE for samples of a batch (batch size =bs). Maximizing this term encourages the output distribution of the generator to be balanced for all the classes.

Batch Statistics Loss. Prior works (Haroush et al., 2020; Yin et al., 2020; Smith et al., 2021) in the centralized setting have recognized that the distribution of synthetic images can drift from real data. We can use batch statistics loss $\mathcal{L}_{BN}$ to avoid such problems. Specifically, the goal is to enforce synthetic images to produce similar statistics in all BatchNorm layers to the ones that are already produced during training. To this end, we minimize the layer-wise distances between the two statistics written as

$$\mathcal{L}_{BN} = \frac{1}{L}\sum_{i=1}^{L} KL(\mathcal{N}(\mu_i, \sigma_i^2), \mathcal{N}(\tilde{\mu}_i, \tilde{\sigma}_i^2)) \tag{3}$$

Here, L denotes the number of BatchNorm layers in the model, μ_i and σ_i are the mean and standard deviation stored in BatchNorm layer i of the global model, $\tilde{\mu}_i, \tilde{\sigma}_i$ are measured statistics of BatchNorm layer i for the synthetic images, KL stands for the Kullback-Leibler divergence.

Finally, we can write the training objective of $\mathcal{G}$ as (4) where w_{div} and w_{BN} control the weight of each term.

$$\min_{\mathcal{G}} \mathcal{L}_{ce} + w_{div}\mathcal{L}_{div} + w_{BN}\mathcal{L}_{BN}, \tag{4}$$

3.2. Client-side: Continual Learning

For client-side training, inspired by (Smith et al., 2021), we distill the *stability-plasticity* dilemma into three critical requirements of CL and aim to address them one by one.

Current task. To have plasticity, the model needs to learn the new features in a way that is least biased toward the old tasks. So, here, the CE loss is computed *for the new classes only* by splitting the linear heads and excluding the old ones:

$$\mathcal{L}_{CE}^{t} = CE(\mathcal{F}_t(x), y) \quad if\ y \in \mathcal{Y}^t\ else\ 0. \tag{5}$$

Previous tasks. To reduce forgetting, we train the model using synthetic and real data simultaneously. However, the distribution of the synthetic data differs from the real one, and it becomes important to prevent the model from distinguishing between old and new data. To address this problem, for fine-tuning the decision boundary using the

	Average Accuracy $\tilde{\mathcal{A}}$ (%)	Average forgetting $\tilde{f}$(%)	Training time (s) ($T=0$)	Training time (s) ($T\geq 1$)	Server Runtime (s)
FedAvg	22.27 ± 0.22	78.77 ± 0.83	≈ 1.2	≈ 1.2	≈ 1.8
FedProx	22.00 ± 0.31	78.17 ± 0.33	≈ 1.98	≈ 1.98	≈ 1.8
FedCIL	26.8 ± 0.44	38.19 ± 0.31	≈ 17.8	≈ 24.5	≈ 2.5 for $T=1$, ≈ 4.55 for $T>1$
FedLwF-2T	22.17 ± 0.13	75.08 ± 0.72	≈ 1.2	≈ 3.4	≈ 1.8
MFCL (Ours)	$\mathbf{43.87 \pm 0.12}$	$\mathbf{28.3 \pm 0.78}$	≈ 1.2	≈ 3.7	≈ 330 (once per task), ≈ 1.8 O.W.
Oracle	67.12 ± 0.4	$--$	≈ 1.2	$\approx 1.2 \times\ T$	≈ 1.8

Table 1. Evaluation on CIFAR-100 dataset.

sampled synthetic data ($\tilde{x} = Sample(\mathcal{G}_{t-1})$), clients freeze the feature extraction part and only update the classification head (represented by $\mathcal{F}_t^*$). This loss can be formulated as

$$\mathcal{L}_{FT}^t = CE(\mathcal{F}_t^*(\tilde{x}), y). \tag{6}$$

Finally, to minimize forgetting, the common method is knowledge distillation over the prediction layer. However, (Smith et al., 2021) proposed *importance-weighted feature distillation*: instead of using the knowledge in the decision layer, they use the output of the feature extraction part of the model (penultimate layer). This way, only the more significant features of the old model are transferred, enabling the model also to learn the new features from the new tasks. This can be written as below where $\mathcal{W}$ is the frozen linear head of the model trained on the last task ($\mathcal{W} = \mathcal{F}_{t-1}^L$).

$$\mathcal{L}_{KD}^t = ||\mathcal{W}(\mathcal{F}_t^{1:L-1}(\hat{x})) - \mathcal{W}(\mathcal{F}_{t-1}^{1:L-1}(\hat{x}))||_2^2, \tag{7}$$

In summary, the final objective on the client side as

$$\min_{\mathcal{F}_t} \mathcal{L}_{CE}^t + w_{FT}\mathcal{L}_{FT}^t + w_{KD}\mathcal{L}_{KD}^t, \tag{8}$$

w_{FT} and w_{KD} determine the importance of each loss term.

3.3. Algorithm $MFCL$

For the first task, clients train the model using the $\mathcal{L}_{ce}$. At the end of training task $t=1$, the server trains the generative model by optimizing (4). Then, the server freezes and saves $\mathcal{G}$ and the global model ($\mathcal{F}_{t-1}$). This procedure repeats for all future tasks, with the only difference being that for $t>1$, the server needs to send the current global model ($\mathcal{F}_t$), the previous task's final model ($\mathcal{F}_{t-1}$) and $\mathcal{G}$ to clients. Since $\mathcal{F}_{t-1}$ and $\mathcal{G}$ are fixed during the whole process of training $\mathcal{F}_t$, the server can send them to each client once per task to reduce the communication cost. To further decrease this overhead, we can use communication-efficient methods, such as (Qiu et al., 2022; Babakniya et al., 2022), that highly compress the model with minor performance degradation.

4. Experiments

Setting. We demonstrate the efficacy of our method on dataset: CIFAR-100 (Krizhevsky et al., 2009). We use the baseline ResNet18 (He et al., 2016) as the global model and ConvNet architecture for $\mathcal{G}$. In our experiments, there are 50 clients in total and 5 randomly sampled participants in every round. Also, there are 10 non-overlapping tasks ($T=10$), and for each task, the model is trained for 100 FL rounds. We use Latent Dirichlet Allocation ($\alpha=1$) (Reddi et al., 2020) to distribute the data of each task among the clients. We compare the baselines based on three metrics –average accuracy, average forgetting and wallclock time– which we explain more in the appendix. All the results are reported after averaging over 3 different random seeds.

Baseline. We compare our method with **FedAvg** (McMahan et al., 2017), **FedProx** (Li et al., 2020), **FedCIL** (Qi et al., 2023), **FedLwF-2T**(Usmanova et al., 2021) and **Oracle**. **FedAvg** and **FedProx** are the two most common aggregation methods in FL. **FedCIL** is a GANs-based method where clients train the discriminator and generator locally to generate samples from the old tasks. In **FedLwF-2T**, clients use two teachers – the global model and their previously trained local model – to distill their knowledge of the past. Finally, **Oracle** as an upper bound: during the training of the i_{th} task, clients have access to all of their data from $t=1,...,i$.

Metrics. We evaluate each approach with the following metrics;

- *Accuracy ($\mathcal{A}^t$)*: Accuracy of the model at the end of task t, over all the classes observed so far.
- *Average Accuracy ($\tilde{\mathcal{A}}$)*: Average of all $\mathcal{A}^t$ for all the T available tasks.

$$\tilde{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{A}^t \tag{9}$$

- *Forgetting (f^t)*: The difference between the highest accuracy of the model on task t and its performance at the end of the training.
- *Average Forgetting ($\tilde{f}$)*: Average of the forgetting over all the tasks.

$$\tilde{f} = \frac{1}{T-1}\sum_{t=1}^{T-1} f^t \tag{10}$$

- *Wallclock time.* This is the time that it takes for the client or server to perform one round of federated learn-

ing. The time is measured in seconds and averaged between different clients and rounds. It is worth noting that all the experiments are done in the same GPU, and the number could change by changing the hardware.

4.1. Results

Table 1 shows each method's average forgetting and accuracies. FedAvg and FedProx have the highest forgetting as they are not designed for FCL. Also, high forgetting for FedLwF-2T indicates that extra teachers cannot be effective in the absence of old data. FedCIL and MFCL have lower forgetting and better accuracy. MFCL outperforms FedCIL because the generative models in FedCIL need to train for a long time to generate effective synthetic data.

We also compare methods' compute costs. Some methods change after learning the first task; therefore, we distinguish between the cost of the first task and later ones. As depicted, MFCL can significantly improve accuracy and forgetting at the cost of a slight increase in the clients' training time for $T > 1$ (due to using synthetic data).

The server cost in MFCL is similar to FedAvg except at the end of each task, where it needs to train the generative model. This extra computation cost should not be a bottleneck because it occurs once per task, and servers usually have access to better computing power compared to clients.

5. Discussion

5.1. Overheads of generative model

Client-side. Using $\mathcal{G}$ on the client side would increase the computational costs compared to vanilla FedAvg. However, existing methods in CL often need to impose additional costs such as memory, computing, or both to mitigate catastrophic forgetting. Nevertheless, there are ways to reduce costs for MFCL. For example, clients can perform inference once, generate and store synthetic images only for training, and then delete them all. They can further reduce costs by requesting that the server generate synthetic images and send them the data instead of $\mathcal{G}$. Here, we raise two crucial points about the synthesized data. Firstly, there is an intrinsic distinction between storing synthetic (or $\mathcal{G}$) and actual data; the former is solely required during training, and clients can delete them right after the training. Conversely, the data in episodic memory should always be saved on the client's side because once deleted, it becomes unavailable. Secondly, synthetic data is shared knowledge that can assist anyone with unbalanced data or no memory in enhancing their model's performance. In contrast, episodic memory can only be used by one client.

Server-side. The server needs to train the $\mathcal{G}$ **once per task**. It is commonly assumed that the server has access to more powerful computing power and can compute more information faster than clients. This training step does not have overhead on the client side and, overall, might slow down the whole process. However, tasks do not change rapidly in real life, giving the server ample time to train the generative model before any shifts in trends or client data occur.

Communication cost. Transmitting the generative model can be a potential overhead for MFCL, as it is a cost that clients must bear **once per task** to prevent or reduce catastrophic forgetting. However, several possible methods, such as compression, can significantly reduce this cost while still maintaining excellent performance. This could be an interesting direction for future research.

5.2. Privacy of MFCL

Federated Learning, specifically FedAvg, is vulnerable to different attacks, such as data poisoning, model poisoning, backdoor attacks, and gradient inversion attacks (Kairouz et al., 2021; Lyu et al., 2020; Fang et al., 2020; Geiping et al., 2020; Chen et al., 2022; Li et al., 2020).

MFCL generally does not introduce any additional privacy issues and is prone to the same set of attacks as FedAvg. MFCL trains the generative model based on the weights of the global aggregated model, which is already available to all clients in the case of FedAvg. On the contrary, in some of the prior work, the clients need to share a locally trained generative model or perturbed private data, potentially causing more privacy problems.

For FedAvg, various solutions and defenses, such as differential privacy or secure aggregation (Wei et al., 2020; Bonawitz et al., 2016), are proposed to mitigate the effect of such privacy attacks. One can employ these solutions in the case of MFCL as well. Particularly, in MFCL, the server **does not** require access to the individual client's updates and uses the aggregated model for training. Therefore, training a generative model is still viable after incorporating these defense mechanisms.

MFCL benefits from Batch Statistics Loss ($\mathcal{L}_{BN}$) in training the generative model. However, some defense mechanisms suggest not sharing local Batch Statistics with the server. While training the generative model without the $\mathcal{L}_{BN}$ is still possible; it can reduce the accuracy. Addressing this is an interesting future direction.

6. Conclusion

This work presents a federated Class-IL framework while addressing resource limitations and privacy challenges. We exploit generative models trained by the server in a data-free fashion, obviating the need for the client's memory.

References

Babakniya, S., Kundu, S., Prakash, S., Niu, Y., and Avestimehr, S. Federated sparse training: Lottery aware model compression for resource constrained edge. *arXiv preprint arXiv:2208.13092*, 2022.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

Chen, C.-L., Babakniya, S., Paolieri, M., and Golubchik, L. Defending against poisoning backdoor attacks on federated meta-learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(5):1–25, 2022.

Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., and Tian, Q. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.

Dong, J., Wang, L., Fang, Z., Sun, G., Xu, S., Wang, X., and Zhu, Q. Federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10164–10173, 2022.

Fang, M., Cao, X., Jia, J., and Gong, N. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pp. 1605–1622, 2020.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Haroush, M., Hubara, I., Hoffer, E., and Soudry, D. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendryx, S. M., KC, D. R., Walls, B., and Morrison, C. T. Federated reconnaissance: Efficient, distributed, class-incremental learning. *arXiv preprint arXiv:2109.00150*, 2021.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jiang, Z., Ren, Y., Lei, M., and Zhao, Z. Fedspeech: Federated text-to-speech with continual learning. *arXiv preprint arXiv:2110.07216*, 2021.

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. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Lyu, L., Yu, H., and Yang, Q. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

Ma, Y., Xie, Z., Wang, J., Chen, K., and Shou, L. Continual federated learning based on knowledge distillation.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Priyanshu, A., Sinha, M., and Mehta, S. Continual distributed learning for crisis management. *arXiv preprint arXiv:2104.12876*, 2021.

Qi, D., Zhao, H., and Li, S. Better generative replay for continual federated learning. *arXiv preprint arXiv:2302.13001*, 2023.

Qiu, X., Fernandez-Marques, J., Gusmao, P. P., Gao, Y., Parcollet, T., and Lane, N. D. Zerofl: Efficient on-device training for federated learning with local sparsity. *arXiv preprint arXiv:2208.02507*, 2022.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., and Zeitak, I. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Smith, J., Hsu, Y.-C., Balloch, J., Shen, Y., Jin, H., and Kira, Z. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9374–9384, 2021.

Usmanova, A., Portet, F., Lalanda, P., and Vega, G. A distillation-based approach integrating continual learning and federated learning for pervasive services. *arXiv preprint arXiv:2109.04197*, 2021.

Van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Vincent Poor, H. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. doi: 10.1109/TIFS.2020.2988575.

Yin, H., Molchanov, P., Alvarez, J. M., Li, Z., Mallya, A., Hoiem, D., Jha, N. K., and Kautz, J. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Yoon, J., Jeong, W., Lee, G., Yang, E., and Hwang, S. J. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889. PMLR, 2021.

A. Algorithm in detail

Algorithm 1 MFCL

N: #Clients, $[\mathcal{C}_N]$: Client Set, K: #Clients per Round, u_i: client i Update, E: Local Epoch
R: FL Rounds per Task, T: #Tasks, t: current task , $|\mathcal{Y}|^t$: Task t Size, q: #Discovered Classes
$\mathcal{F}_t$: Global Model for task t, $\mathcal{G}_t$: Generative Model, $E_{\mathcal{G}}$: Generator Training Epoch
$q \leftarrow 0$
$\mathcal{G}, \mathcal{F}_0, \mathcal{F}_1 \leftarrow$ **initialize**()
for $t = 1$ **to** T **do**
  $q \leftarrow q + |\mathcal{Y}^t|$
  $\mathcal{F}_t \leftarrow$ **updateArchitecture**$(\mathcal{F}_t, q)$
  for $r = 1$ **to** R **do**
    $C_K \leftarrow$ **RandomSelect**$([\mathcal{C}_N], K)$
    for $c \in C_K$ in parallel **do**
      $\mathcal{U}_c \leftarrow$ **localUpdate**$(\mathcal{F}_t, \mathcal{G}, \mathcal{F}_{t-1}, E)$
    end for
    $\mathcal{F}_t \leftarrow$ **globalAggregation**$(\mathcal{F}_t, [\mathcal{U}_c])$
  end for
  # save a frozen version of model for sending to clients
  saveFrozen$(\mathcal{F}_t)$
  $\mathcal{G} \leftarrow$ **trainDFGenerator**$(\mathcal{F}_t, E_{\mathcal{G}}, q)$ #using (4)
  $\mathcal{G} \leftarrow$ **freezeModel**$(\mathcal{G})$ #fix generator weights
end for

A.1. Generative Model Architectures

In Table 2, we show the generative model architectures used for CIFAR-100. The global model has ResNet18 architecture, we change the first CONV layer kernel size to 3×3 from 7×7. In this table, CONV layers are reported as CONV$K \times K(C_{in}, C_{out})$, where K, C_{in} and C_{out} are the size of the kernel, input channel and output channel of the layer, respectively.

A.2. Hyperparameters

Table 3 presents some of the more important parameters.

CIFAR-100
FC$(1000, 128 \times 8 \times 8)$
reshape$(-, 128, 8, 8)$
BatchNorm(128)
Interpolate(2)
CONV$3 \times 3(128, 128)$
BatchNorm(128)
LeakyReLU
Interpolate(2)
CONV$3 \times 3(128, 64)$
BatchNorm(64)
LeakyReLU
CONV$3 \times 3(64, 3)$
Tanh
BatchNorm(3)

Table 2. Generative model Architecture

Dataset	**CIFAR-100**
Data Size	32×32
# Tasks	10
# Classes per task	10
# Samples per class	500
Batch Size	32
Synthetic Batch Size	32
FL round per task	100
Local epoch	10

Table 3. Parameter Settings in different datasets