TU Delft

Delft University of Technology

Parameterizing Federated Continual Learning for Reproducible Research

Cox, Bart; Galjaard, Jeroen; Shankar, Aditya; Decouchant, Jérémie; Chen, Lydia Y.

DOI
10.1007/978-3-031-74643-7_35

Publication date
2025

Document Version
Final published version

Published in
Machine Learning and Principles and Practice of Knowledge Discovery in Databases

Citation (APA)
Cox, B., Galjaard, J., Shankar, A., Decouchant, J., & Chen, L. Y. (2025). Parameterizing Federated Continual Learning for Reproducible Research. In R. Meo, & F. Silvestri (Eds.), *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2023, Turin, Italy, September 18–22, 2023, Revised Selected Papers, Part V* (pp. 478-486). (Communications in Computer and Information Science; Vol. 2137 CCIS). Springer. https://doi.org/10.1007/978-3-031-74643-7_35

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Parameterizing Federated Continual Learning for Reproducible Research

Bart Cox(✉), Jeroen Galjaard, Aditya Shankar, Jérémie Decouchant, and Lydia Y. Chen

Delft University of Technology, Delft, The Netherlands
B.A.Cox@tudelft.nl

Abstract. Federated Learning (FL) systems evolve in heterogeneous and ever-evolving environments that challenge their performance. Under real deployments, the learning tasks of clients can also evolve with time, which calls for the integration of methodologies such as Continual Learning (CL). To enable research reproducibility, we propose a set of experimental best practices that precisely capture and emulate complex learning scenarios. To the best of our knowledge, our framework, `Freddie`, is the first entirely configurable framework for Federated Continual Learning (FCL), and it can be seamlessly deployed on a large number of machines leveraging containerization and Kubernetes. We demonstrate the effectiveness of `Freddie` on two use cases, (i) large-scale concurrent FL on CIFAR100 and (ii) heterogeneous task sequence on FCL, which highlight unaddressed performance challenges in FCL scenarios.

Keywords: Federated Continual Learning · Resource and Data Heterogeneity · Reproducible Research

1 Introduction

Federated Learning (FL) [14] performs distributed optimization thanks to a central federator server that maintains a global model using model updates computed by clients. It is common for data to be distributed among the clients of an FL system in a non-independent and identically distributed (non-IID) way. Moreover, in practice, client learning tasks also evolve over time. Continual Learning (CL) [5] is a technique that addresses the scenario where a model is continuously trained on evolving client tasks.

One of the key challenges in CL is catastrophic forgetting: parameters or semantic representations learned for past tasks drift under the influence of new tasks. Three categories of techniques address this challenge [7]. Replay mechanisms, like Gradient Episodic Memory (GEM) [10] and Deep Generative Replay [18], retain or generate data from earlier tasks for new task adaptation, which allows the network to revise previously learned tasks. Regularization techniques, such as Elastic Weight Consolidation (EWC) [5], penalize the divergence of model parameters, preventing the adaptation process on new tasks from deviating too far from the model learned on prior tasks. Parameter isolation methods

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
R. Meo and F. Silvestri (Eds.): ECML PKDD 2023, CCIS 2137, pp. 478–486, 2025.
https://doi.org/10.1007/978-3-031-74643-7_35

use specific weights of the network for the task at hand, i.e., use a mask to freeze the weights of other tasks [12].

Continual Learning allows a client to learn from its previous tasks if features are repeated over time. Federated Continual Learning [21] (FCL) combines CL and FL, enabling clients to indirectly learn from each other. Existing CL frameworks do not take this indirect learning into account and therefore provide limited support for Federated Continual Learning.

In addition, reproducing FCL results that were obtained in deployment is difficult. For example, experimental environments are often tightly controlled and steady, while real-world environments are often dynamic and heterogeneous. In addition, clients might be punctually busy processing co-located tasks. Several FL simulation [11,17] and emulation [1,17] frameworks have been proposed, but they cannot be easily extended to support heterogeneous data, learning tasks and hardware platforms. In addition, frameworks that focus on enabling large-scale FL experiments impose a significant overhead to manage the execution or require the use of a strict pipeline. In this paper, we address the lack of a scalable yet flexible framework for reproducible FCL experiments. Overall, we make the following contributions:

- We identify key requirements for scientific FL and FCL emulation: ease of use, reproducibility, support for complex workloads, and resource heterogeneity.
- We develop `Freddie`—a framework for Federated and distributed machine-learning— to the best of our knowledge, the first open source[1] framework that addresses these requirements. `Freddie` supports small scale deployments, i.e., single machine simulations, and large-scale emulation over self-managed and cloud systems using containerization and Kubernetes. `Freddie` enables the emulation of both data and resource heterogeneity.
- We provide benchmarking generating methods for FCL that explore both data and task heterogeneity across clients—realistic workloads tailored for Federated Continual Learning systems.

2 Related Work

Federated Learning (FL). Existing FL frameworks support a fixed set of learning tasks across clients during training. Flower [1] provides a client-server framework that needs to be manually started on different devices. Differently, Fate [3] focuses on providing a secure and production-ready Federated Learning setup. Fate supports Kubernetes deployments but requires the use of its pipelines to run experiments. Although this provides desirable security additions for production systems, it tends less to prototyping and active research needs. Besides research endeavors, popular deep learning platforms such as TorchX and Tensorflow Federated can respectively run distributed and Federated experiments at scale, but they lack the flexibility to use other ML libraries.

[1] https://gitlab.ewi.tudelft.nl/dmls/publications/freddie.

Continual Learning (CL). FACIL [13], PyCIL [22], and Pycontinual [4] provide CL frameworks and CL algorithms such as Learning without Forgetting [8], incremental Classifier and Representation Learning [16], EWC, and GEM. Continual World [20] adds a simulation world for robotics tasks for Continual Reinforcement Learning. Avalanche [9] is focused on reproducible End-to-End Continual Learning. The aforementioned frameworks support CL only on a single machine. FedWEIT [21] combines parameter isolation and regularization and extends CL to a Federated setting. However, it does not consider the impact of task sequences on the global model's quality. Lastly, current FL frameworks cannot be easily extended to support CL scenarios where the output types evolve.

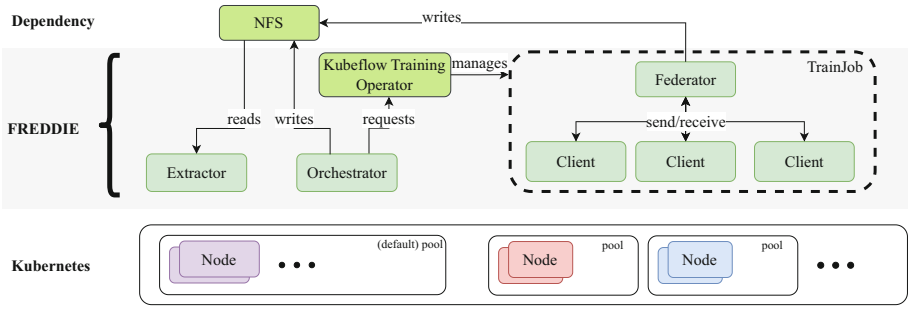


Fig. 1. Overview of `Freddie`. An Orchestrator and an Extractor are used for deploying experiments and collecting data. Experiments are run as TrainJobs managed by Kubeflow Training Operators. Within such a job, the defined job is run using a federator and one or more clients with local data.

3 `Freddie`: A Framework for Reproducible FCL Research

We first provide a brief overview of system requirements addressed by `Freddie`. Following, due to space limitations, we focus on `Freddie`'s implementation leveraging containerization and orchestration methods and its support for FCL.

Classical FL Parameters. The number of clients, the federators' aggregation, and client selection strategies must be configurable. In addition, FL-related and common hyperparameters, such as the training epochs, learning rates, etc., must be configurable.

Statistical Heterogeneity. As in FL, local data distributions remain highly important, and their non-iidness should be configurable. In the context of FCL, local distributions also limit the tasks that clients might be able to train for.

Resource Heterogeneity. It should be possible to specify the computing power of the clients and the federator and the characteristics (latency, throughput) of the network links that interconnect them.

Task Definition. For FCL, the task sequence of each client can be specified in conjunction with statistical heterogeneity. Non-IID task distributions can be assimilated to the situation where clients learn tasks with high intra-task variance, e.g. due to different domains. In such settings, it is often unclear how the quality of current CL methods is impacted by aggregation. Testing of a CL task can be done in multiple ways: Task- Incremental Learning (task-IL) assumes task IDs are present during testing, while Domain-Incremental Learning (domain-IL) [19] drops this assumption. Both methods are supported by `Freddie`.

Kubernetes and Containers. To allow for experimental evaluation of FCL, experiments should be deployable on both single and multiple systems. `Freddie` leverages containerization of all FL nodes to provide flexible and scalable deployments. Figure 1 shows a high-level overview of `Freddie` deployed in a Kubernetes cluster. The Orchestrator starts and manages experiments within a Kubernetes cluster, leveraging Kubeflow's [6] training operators. The Orchestrator provided with experiment configurations monitors resources and schedules the execution of the experiment. The Extractor provides storage for experiments and an access point to retrieve logs and artifacts. federator and client nodes then execute the Federated (Continual) Learning experiment. The federator and client provide flexible and extensible interfaces for users to extend and use for experiments. `Freddie` provides basic implementations for users to extend or adapt.

The overall flow of an FCL experiment with `Freddie` is as follows. After the user submits a configuration, the Orchestrator deploys and manages the execution within a cluster. The federator and clients are automatically configured within the deployed experiment, allowing the experiment to start after all parties come online. The communication between any two nodes in the system is asynchronous, allowing the development of FL systems with non-blocking federator-client interactions. Finally, the Extractor allows users to store and retrieve experiment statistics and artifacts created by the federator or clients. This design allows users to scale their experiments up from small-scale prototyping with minimal effort using Kubernetes or run experiments containerized or locally.

Novel Support for FCL. `Freddie` supports the SOTA algorithms for FCL [21] and common CL methods such as EWC and GEM. For CL, `Freddie` implements Task-IL and Domain-IL [19] through sliding, expanding, and full window mechanisms. A sliding window restricts the output classes only to those of the task evaluated at a time t. Expanding windows do not utilize the task IDs to make any such restriction, so the output classes include all classes learned until that time.

Table 1. System and hyperparameters used in 'small' and 'scale' experiments. All experiments were run on 'e2-standard-8' nodes.

	System				Federator (F)			Clients (C)		
	Nodes	#C	CPU (F/C)	Memory (F/C)	Strategy	#Rounds (R)	#C/R	Model	Data	BS
Small	2,2,3	5,10,20	2/1	2/2G	FedAvg	100	5	LeNet	CIFAR10	64
Scale	4,12	25,75		2/2,6G		85	all	ResNet	CIFAR100	

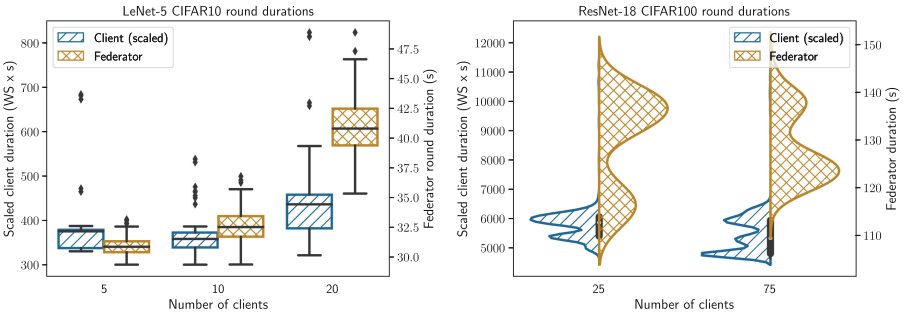
A full window does not restrict outputs based on the task and can be used in the standard Federated Learning scenario. The added complexity of FCL allows for workloads over the same set of tasks to produce different results. We devise three schemes that partition tasks differently, and that can be used to evaluate an FCL scheme over a representative set of scenarios. We discuss these three schemes: Column, Shuffled and Balanced. *Column* splits the CL workload such that all clients handle all tasks in the same order, increasing the expected catastrophic forgetting effect. *Shuffled* randomly generates task orderings in which to learn tasks across clients, thereby relying on pseudo-randomness in conjunction with a pre-specified seed. *Balanced* staggers the task ordering across clients so that tasks are seen in consecutive rounds by different clients. This task ordering scheme allows for evaluation in FCL with low expected short-term forgetting, while long-term forgetting may still occur. `Freddie`s' provides CL data wrappers for Federated datasets, providing a flexible way to define non-IID FCL datasets.

4 Performance Evaluation

We demonstrate some features of `Freddie` through experiments. For this purpose, we use the overlapping CIFAR100 dataset. The original targets of CIFAR100 are used to partition the data into different tasks for FCL, following the same steps as in [21]. We first consider a FL scenario with the default version of CIFAR100. Following we consider the overlapping CIFAR100 split into 10 separate tasks in a FCL scenario. We use the average accuracy metric following the CL literature [2,15].

Scalability. To investigate `Freddie`'s emulation capability, we perform a *small* and *large* scale experiment on a Google Kubernetes Engine (GKE) cluster to cover possible use cases. During deployment, the pods of the federator and clients were run on a separate node pool scaled to meet each experiment's requirements. We study the performance of an FL experiment emulated on a CPU-enabled Kubernetes cluster, where multiple clients may run on a single Kubernetes node. Parameters of the experiments are provided in Table 1.

The *small* experiment in Fig. 2a depicts the spread round times of clients (scaled) and the federator, with 5 selected clients per round. The client round duration is scaled by the number of clients (World Size WS) ($|\mathcal{D}_{Cifar}|$/WS)



(a) **Small:** round duration over 100 rounds with 5 clients per round.

(b) **Scale:** round duration distribution with a participation rate of 1 (n=3).

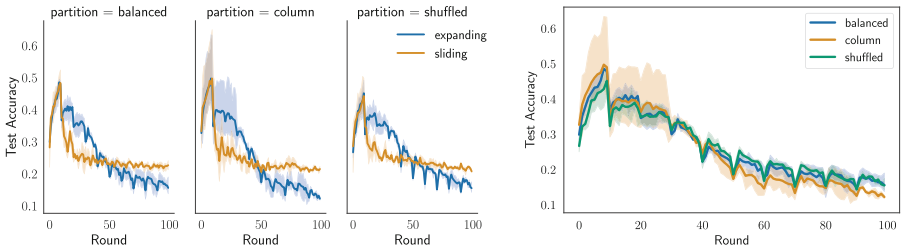
Fig. 2. Client (scaled) and federator round duration with `Freddie` for 'small' (LeNet5 & CIFAR10) and 'scale' experiment (ResNet-18 & CIFAR100).

to account for differences in clients' datasets as the WS increases. The outliers in the plots originate from the first epoch run on clients, which are inherently slower due to loading data into memory. Nevertheless, it is expected that the scaled client duration stays relatively constant, while the result shows an increase as the number of clients increases (from 115 to 123 s, and from 136 to 138 s). Similarly, the federator sees a positive correlation between round duration and WS. The number of co-scheduled clients on the same node can explain this trend, as the networking overhead stays the same.

For the *scale* experiments, we provide the round time density estimate in Fig. 2b. The client round times exhibit the same range of processing times that were observed in the 'small' setting. In both settings, participating clients in each round may run on the same node, varying from 3 to 7 clients per node. We use similar settings in the 'small' configuration that involves 20 clients, where 4 nodes are used. As such, confirming that resource contingencies due to co-scheduling will likely cause the increased client round time with 20 clients. The different modes within the client's round duration can be explained by imperfect data splits and the imbalanced assignment of the number of clients to be co-scheduled with the federator. The federator's density estimate shows a similar pattern with two distinct modes. With the cluster configurations employed, i.e., 4 and 12 nodes, it is possible for the federator to be co-scheduled on a machine with different numbers of clients. As a result, the federator experiences a variable level of resource contingency. However, an increase in the two modes is visible as the number of clients increases, which is expected due to the increased communication volumes.

Task-IL vs. Domain-IL. Assuming that the model can pre-condition on the ID of the task it is currently training on, or evaluating, increases its accuracy. For FCL, Task-Incremental Learning and Domain-Incremental Learning are imple-

mented using the sliding and expanding window, respectively. Let us recall that sliding windows use task IDs, contrary to expanding windows. For the overlapping CIFAR100 dataset, if one assumes that the task ID is known, then the number of output classes is restricted to only the five subclasses within that task. Thus leading to higher average task probabilities for Task-IL scenarios. This difference is prevalent in Fig. 3a. Under the expanding window scheme, classification outputs one of $5T$ classes, where T is the number of tasks learned until evaluation time. Therefore, the probability of classifying correctly is even lower than in the sliding window scenario. Figure 3a shows the positive impact of leveraging the task ID on accuracy. Using sliding-window results in higher accuracy than expanding-window, which sometimes has to be used because of the application use case. Because of this difference, `Freddie` supports both Task-IL and Domain-IL.



(a) Decreased forgetting under Task-IL vs. Domain-IL due to task-awareness.

(b) Domain-IL impact of task distribution across clients over time.

Fig. 3. Federated Continual Learning impact of task awareness and task order.

FCL Task Heterogeneity. As discussed in Sect. 3, tasks can be processed in different orders at each client. To demonstrate the impact of different task distributions over time, we implement the Overlapped-CIFAR100 dataset with 20 tasks that can be used for FCL [21]. The accuracy in Fig. 3b is calculated as the average accuracy of all tasks seen until that point, resulting in expected 'drops' in accuracy as new tasks are introduced. Indeed, the learning curves in Fig. 3b shows noticeable drops over time. However, different trends are visible between workloads. The column scheme suffers more from more pronounced *catastrophic forgetting* than the shuffled and balanced scheme, resulting in lower accuracy. We observe that the column scheme results in an average 4% test accuracy drop compared to the shuffled and partition schemes.

5 Conclusion

We presented `Freddie`, the first framework for reproducible Federated Continual Learning research, which is motivated by the increasing importance of Federated and Continual Learning. `Freddie`'s deployment abilities on different platforms, scalability with the number of clients, and support for data and task heterogeneity provide FL practitioners with a powerful tool. Our experimental results showcase previously unaddressed performance issues that Federated Continual Learning systems might face: severe catastrophic forgetting in different task heterogeneity settings. `Freddie` is open-source. Future work consists of supporting new CL datasets, algorithms, and generative models.

References

1. Beutel, D.J., Topal, T., Mathur, A., et al.: Flower: a friendly federated learning research framework. CoRR arxiv:2007.14390 (2020)
2. Chaudhry, A., Ranzato, M., Rohrbach, M., et al.: Efficient lifelong learning with A-GEM. In: ICLR (2019)
3. FedAI: Fate (2019). https://fate.fedai.org/
4. Ke, Z., Liu, B., Ma, N., et al.: Achieving forgetting prevention and knowledge transfer in continual learning. In: NeurIPS (2021)
5. Kirkpatrick, J., Pascanu, R., Rabinowitz, N.C., et al.: Overcoming catastrophic forgetting in neural networks. CoRR arxiv:1612.00796 (2016)
6. Kubeflow: Kubeflow (2018). https://kubeflow.org
7. Lange, M.D., Aljundi, R., Masana, M., et al.: Continual learning: a comparative study on how to defy forgetting in classification tasks. CoRR arxiv:1909.08383 (2019)
8. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. **40**(12), 2935–2947 (2018)
9. Lomonaco, V., Pellegrini, L., Cossu, A., et al.: Avalanche: an end-to-end library for continual learning. In: CVPR (2021)
10. Lopez-Paz, D., Ranzato, M.A.: Gradient episodic memory for continual learning. In: NeurIPS (2017)
11. Ma, Y., Yu, D., Wu, T., et al.: PaddlePaddle: an open-source deep learning platform from industrial practice. Front. Data Comput. **1**(1), 105–115 (2019)
12. Mallya, A., Lazebnik, S.: Packnet: adding multiple tasks to a single network by iterative pruning. In: CVPR (2018)
13. Masana, M., Liu, X., Twardowski, B., Menta, M., et al.: Class-incremental learning: survey and performance evaluation on image classification. IEEE Trans. Pattern Anal. Mach. Intell. **45**(5), 5513–5533 (2023)
14. McMahan, B., Moore, E., Ramage, D., et al.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
15. Mirzadeh, S.I., Farajtabar, M., Pascanu, R., et al.: Understanding the role of training regimes in continual learning. In: NeurIPS (2020)
16. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: incremental classifier and representation learning. In: CVPR (2017)
17. Reina, G.A., Gruzdev, A., Foley, P., et al.: Openfl: an open-source framework for federated learning. CoRR arxiv:2105.06413 (2021)

18. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NeurIPS, pp. 2990–2999 (2017)
19. van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. CoRR arxiv:1904.07734 (2019)
20. Wołczyk, M., Zajkac, M., Pascanu, R., et al.: Continual world: a robotic benchmark for continual reinforcement learning. In: NeurIPS (2021)
21. Yoon, J., Jeong, W., Lee, G., et al.: Federated continual learning with weighted inter-client transfer. In: ICML (2021)
22. Zhou, D.W., Wang, F.Y., Ye, H.J., et al.: Pycil: a python toolbox for class-incremental learning. CoRR arxiv:2112.12533 (2021)