Data Heterogeneity and Forgotten Labels in Split Federated Learning

Joana Tirana[1*], **Dimitra Tsigkari**[2], **David Solans Noguero**[2], **Nicolas Kourtellis**[3]

[1]University College Dublin, Ireland
[2] Telefónica Scientific Research, Barcelona, Spain
[3]Keysight AI Labs, Barcelona, Spain
joana.tirana@ucdconnect.ie, {dimitra.tsigkari, david.solansnoguero}@telefonica.com, nicolas.kourtellis@keysight.com

Abstract

In Split Federated Learning (SFL), the clients collaboratively train a model with the help of a server by splitting the model into two parts. Part-1 is trained locally at each client and aggregated by the aggregator at the end of each round. Part-2 is trained at a server that sequentially processes the intermediate activations received from each client. We study the phenomenon of catastrophic forgetting (CF) in SFL in the presence of data heterogeneity. In detail, due to the nature of SFL, local updates of part-1 may drift away from global optima, while part-2 is sensitive to the processing sequence, similar to forgetting in continual learning (CL). Specifically, we observe that the trained model performs better in classes (labels) seen at the end of the sequence. We investigate this phenomenon with emphasis on key aspects of SFL, such as the processing order at the server and the cut layer. Based on our findings, we propose Hydra, a novel mitigation method inspired by multi-head neural networks and adapted for the SFL setting. Extensive numerical evaluations show that Hydra outperforms baselines and methods from the literature.

Code — https://github.com/jtirana98/Hydra-CF-in-SFL
Extended version — https://arxiv.org/abs/2511.09736

1 Introduction

Split Learning (SL) is a distributed learning method where part of the clients' training is offloaded to a server (Vepakomma et al. 2018). This is particularly useful in cases where clients' resources are insufficient to perform on-device training. In a nutshell, in SL, the deep neural network (NN) model is split into two parts of consecutive layers: part-1 and part-2, where the last layer of part-1 is called the cut layer. These parts are trained by the client and the server, respectively. There is a plethora of proposed variations for SL that mostly differ in the way the training at the server takes place. In this work, we focus on Split Federated Learning (SFL) and, in particular, SplitFedv2 (Thapa et al. 2022), which is one of the most prevailing versions (Hafi et al. 2024). In fact, thanks to its workflow, SFL has been shown to deliver a faster convergence than other variants (Thapa et al. 2022).

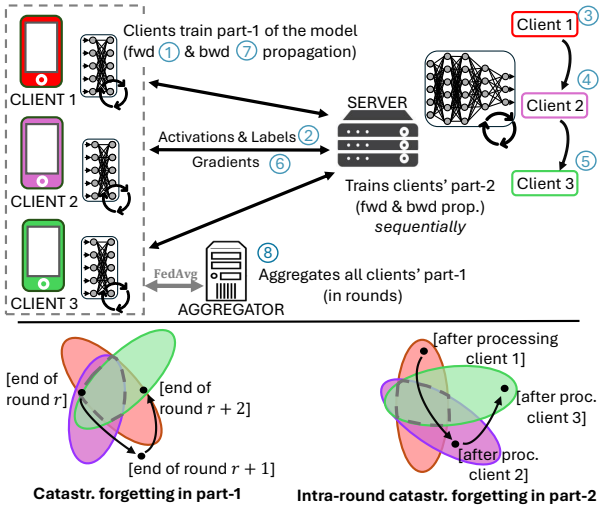

Figure 1: (Top) The steps of the processing workflow of SFL. (Bottom) Parameter spaces for low error for clients' dominant labels (under data heterogeneity), where a client's color represents its dominant label. Part-1 of the model suffers from catastrophic forgetting from round to round. Part-2 suffers from intra-round catastrophic forgetting due to the processing order at the server.

The processing workflow of SFL for 3 clients is depicted in Fig. 1 (top). At the beginning of every round, the clients perform forward propagation of part-1 of the model on a batch of their samples (step 1 in Fig. 1) and transmit the activations of the cut layer to the server, along with the labels of the batch (step 2). Then, the server sequentially trains part-2 for each client, propagating forward and backward (steps 3-5), and transmits the gradients to the clients (step 6). Based on the received gradients, the clients perform backward propagation of part-1 (step 7). Steps 1 to 7 are repeated for all the batches, and once all clients have processed all their data, the aggregator aggregates the local version of part-1 and sends a global part-1 to the clients (step 8), before a new round begins. We consider here the typical case where clients' data remains unchanged throughout the training.

Motivation. Part-1 is trained as in Federated Learning (FL), while part-2 is trained over the activations that clients send.

[*]Work completed while at Telefónica Scientific Research.

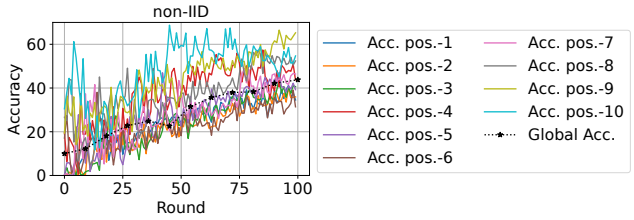

Figure 2: Accuracy per processing position (at the server) and global accuracy achieved by MobileNet in CIFAR-10 under non-IID data distributions among 10 clients.

This *dual aspect* of SFL makes it particularly susceptible to catastrophic forgetting (CF) in the presence of data heterogeneity, a common phenomenon in training NNs (Goodfellow et al. 2013). In practice, part-1 and part-2 may suffer from CF differently, as depicted in Fig. 1 (bottom). Like in FL, the local updates of part-1 drift away from the global optima as a result of the aggregation (e.g., `FedAvg` (McMahan et al. 2017)) at the end of every round. On the other hand, our experiments reveal a new finding: the processing order at the server has a significant impact on CF in part-2. This CF resembles the one of Continual Learning (CL) (Wang et al. 2024): the model parameters perform well for the dominant label of the last processed client, but forget the previously seen labels, i.e., in every update step at the server, part-2 tends to forget the knowledge learned during the previous steps, leading to a *performance gap* among labels.

Despite the resemblance to forgetting in CL, there are several differences between SFL and CL. Notably, in SFL, the data distribution is fixed and the data stream is finite, i.e., the data is not temporal. Further, two types of repetitions occur (not to be confused with knowledge replay): (i) all clients' input (i.e., part-1's activations) is cyclically passed through the server at each round, and (ii) depending on the processing order of the clients' part-2, input concerning a specific label may be repeated within the same round.

Fig. 2 showcases the CF in SFL as a result of the processing order at the server under non-independently and identically distributed (non-IID) data. In particular, we assume that the data of each client contains a highly represented (dominant) label. Each line depicts the accuracy of the labels that are dominant in clients that are processed at a certain position (1 to 10, in CIFAR-10 with 10 labels), where details on the experimental setup will be given in Sec. 3.1. We see that, under non-IID data, the labels that are dominant at the clients who are processed last at the server consistently outperform the other labels. The disparity in performance between the labels seen at the end of the sequence and those seen earlier is related to the CF in part-2. We identify this as *intra-round catastrophic forgetting* (intraCF), as it occurs at a different granularity (i.e., within a round) than the model drifting of part-1, which also contributes to the phenomenon. Moreover, the global accuracy is heavily affected, i.e., 43% after 100 rounds, whereas, under IID data, the global accuracy reaches 80% after 100 rounds.

Challenges. Despite the plethora of works on CF in other settings (CL, FL) and some recent findings in SL and SFL (further discussed in Sec. 2), we identify several challenges.
(i) *Metrics of forgetting & Mitigation Methods.* Related work on forgetting in SL, e.g., (Feng et al. 2023; Xia et al. 2025) focuses solely on the global accuracy, ignoring the disparity (gap) of performance among labels. Further, methods proposed to mitigate forgetting and/or data heterogeneity in CL and FL, such as EWC (Kirkpatrick et al. 2017) and Scaffold (Karimireddy et al. 2020), cannot be applied in the setting of SFL as the former is based on the notion of temporal data and the latter requires the entire model.
(ii) *The roles of the cut layer and processing order.* The majority of related work assumes that the server processes the clients' data in a random order (Thapa et al. 2022; Liao et al. 2024) and often ignores the role of the cut layer in CF. Moreover, existing theoretical analyses on SFL are limited to convergence analysis under standard assumptions without focusing on these key elements of SFL (Han et al. 2024).
(iii) *Empirical evaluation.* The existing theoretical findings are often showcased in limited evaluation scenarios (e.g., partition methods), see (Li and Lyu 2023; Han et al. 2024). Further, it is unclear whether some of the theoretical claims proved in the settings of CL and FL may hold in SFL.
Contributions. The challenges above further corroborate the need for an empirical evaluation of the phenomenon of catastrophic forgetting in SFL and for an efficient mitigation method. This manuscript addresses these challenges by:
1. Identifying a new case of CF in SFL as a result of the processing workflow of SFL.
2. Providing valuable insights into this phenomenon (e.g., w.r.t. the impact of cut layer and processing order at the server) and comparing them with existing theoretical findings. To this end, we focus on the accuracy, the performance gap among labels, and the metric of backward transfer.
3. Proposing Hydra, a novel mitigation method that trains multiple versions of the last layers of part-2 after grouping the clients' input based on their data distributions. Hydra is designed based on the insights from our analysis and induces minimal overhead (which can be further reduced).
4. Showing how Hydra reduces CF and outperforms other methods through numerical evaluations in a variety of NN models, datasets, and data partitions.

2 Related Work

In this section, we discuss related work on data heterogeneity and CF in FL and SL. In FL under non-IID data, several works study CF as a result of model drifting and client selection policies. Mitigation methods utilize knowledge distillation (Lee et al. 2022; Ma et al. 2022) and regularization (Xu et al. 2022), among others. We notice, however, that most of the methods for FL require control of the entire model, e.g., (Zhao et al. 2018; Karimireddy et al. 2020; Hu et al. 2024), which is incompatible with the SFL workflow. Moreover, optimization techniques in FL (Li et al. 2020; Wang et al. 2020) assume a symmetric training workflow with full local steps or per-client regularization terms, which is also not applicable for SFL. On the other hand, replay-based methods for FL, e.g., (Qi, Zhao, and Li 2023; Li et al. 2024) induce computational and memory overhead. Concerning SL (in its various definitions), CF has been mostly studied in

terms of global accuracy under non-IID data. Proposed solutions are based on generated data and knowledge transfer for SplitNN (Feng et al. 2023, 2024), as well as on knowledge replay for SplitFedv1 (Xia et al. 2025). Further, (Madaan, Kulkarni, and Pant 2022) provides a preliminary study of the impact of the processing order in different variants of SL. Finally, the only work that addresses data heterogeneity but not CF in SFL (SplitFedv2) is (Liao et al. 2024), which merges features and adjusts the batch size.

3 Exploring CF in SFL

The goal is to train a (deep) classification model whose set of labels (classes) is $\mathcal{L}$, where $|\mathcal{L}| = L$. The training with a set $\mathcal{C}$ of clients follows the SFL workflow described in Sec. 1. Further, clients' data can be seen as "tasks" (in the parlance of CL) with possibly (totally) overlapping labels.

3.1 Methodology of Experiments

Models, Datasets, and Cut Layer in SFL. Two ML models are employed: MobileNetV1 (Howard et al. 2017), known for its efficiency on mobile and resource-constrained devices, and ResNet101 (He et al. 2016), which is renowned for its depth and robust feature extraction capabilities. These models are trained on SVHN (Netzer et al. 2011), CIFAR-10 ($L = 10$), CIFAR-100 (Krizhevsky, Hinton et al. 2009) ($L = 20$ or 100 – unless otherwise specified, we use 20 superclasses as in (Ramasesh, Dyer, and Raghu 2021)), and TinyImageNet (Le and Yang 2015) ($L = 200$). The cut layer determines the sizes of part-1 and part-2. A shallow cut layer implies a small part-1, i.e., the server trains the model's largest portion, while a deep one implies a small part-2.

Based on the discussion in Sec. 1 about the intraCF, unless otherwise specified, we choose a shallow cut layer, e.g., at layer 4 (out of 26 in total) in MobileNetV1, and layer 2 (out of 35 in total) in ResNet101, as is common in related work (Thapa et al. 2022). In fact, in Sec. 3, we show that a shallow cut better captures the impact of intraCF on the forgetting observed in SFL. We stress, however, that we also perform a sensitivity analysis with respect to the choice of the cut layer, while following the bottleneck approach (Kang et al. 2017). Finally, all the experiments are repeated at least 10 times for 100 training rounds (with varying random seeds) and, then, for each metric (e.g., global accuracy), the median of the observed values from all the repeated experiments is computed. This is to ensure that our results are robust against outliers (Leys et al. 2013).

Data Partitioning and Number of Clients. We use three common data partitioning techniques for data heterogeneity. First, we employ the *Dominant Label (DL) ratio* method, an instance of overlapping label distribution techniques (Jimenez G. et al. 2024). This uses the parameter $p \in [0, 100]$ to control the percentage (i.e., $p\%$) of samples with a dominant label at each client, while the remaining samples (i.e., $(100-p)\%$) are distributed evenly among the other clients. Clearly, as p increases, greater heterogeneity is produced. Throughout the paper, the notation $p\%$-DL is employed, and we use the parameter $\phi \in \mathbb{N}$ to describe the number of clients that, for each label, have this label as dominant. Hence, the total number of clients participating in the training is $C = |\mathcal{C}| = \phi \cdot L$. We also employ the *Sharding* and *Dirichlet* data partitioning methods.

Metrics for Catastrophic Forgetting in SFL. A common metric of forgetting in CL is the *backward transfer* (BW), which has been adapted for FL as follows (Lee et al. 2022):

$$\mathcal{BW} = \frac{1}{L}\sum_{l=1}^{L} \max_{r \in \{1,\ldots,R-1\}} (A_l^r - A_l^R), \tag{1}$$

where A_l^r is the accuracy of label l at round $r < R$. It captures the difference between the maximum ("peak") accuracy and the final accuracy of each label at the end of training (i.e., round R), averaged over all labels. The BW metric is better suited for *dynamic input data*, where the performance of a label may decrease with time as other labels appear more often (e.g., in temporal data). For this reason, and motivated by what we observe in Fig. 2, we also measure the (average) *performance gap (PG)* between labels, for each round r, as follows:

$$\mathcal{PG}(r) = \frac{1}{L}\sum_{l=1}^{L} \max_{k \in \mathcal{L}} \{|\min(0, (A_l^r - A_k^r))|\}. \tag{2}$$

The quantity above measures the maximum difference in performance between each label and the best-performing one, averaged over all labels. Essentially, $\mathcal{PG}(r)$ captures the average "gap" between the per-label accuracies at round r.

We evaluate CF with the two metrics above, while, in Sec. 5, we employ an additional metric in the case of cyclic order and DL partitioning, the *per-position accuracy*. In practice, position-k accuracy, for $k \in \{1..L\}$, is the accuracy of the labels that are dominant in clients that are processed at position k (in the cycle of length L) at the server.

3.2 Insights into Catastrophic Forgetting in SFL

In what follows, we study CF in SFL, while occasionally focusing on intraCF. We first introduce two different types of processing orders at the server.

• **No Specific Order (random).** It is often implied in the literature that the server processes the data received by each client in a random or a first-come-first-served (FCFS) order (Thapa et al. 2022), as the arrival time may depend on network conditions (Tirana et al. 2024). To this end, we simulate the case of FCFS with a *random* function at each round.

• **Cyclic Order.** This order guarantees *structure* in the way the server processes the input of different clients (at part-2). Similar to (Swartworth et al. 2023), at each round, the server processes the clients' input in a cycle of length L based on the label that is highly represented in their data, i.e., first all ϕ clients whose highly represented label is $X \in \mathcal{L}$, then label $Y \in \mathcal{L}$, and so on. This sequence remains the same across all rounds. In our experiments, to ensure that the results are independent of the type of label, the order (sequence) of the labels is randomly selected in each experiment.

Scale of clients (ϕ). In the cyclic order, larger values of ϕ imply that the data of more clients of the same type (e.g., with the same dominant label in DL partition) are processed

Model	DL ratio	Global Accuracy (↑) $\phi_{=1}$	$\phi_{=5}$	$\phi_{=10}$	random	Performance Gap (PG) (↓) $\phi_{=1}$	$\phi_{=5}$	$\phi_{=10}$	random	Backward Transfer (BW) (↓) $\phi_{=1}$	$\phi_{=5}$	$\phi_{=10}$	random
		CIFAR-10											
MobileNet	80	**45**±0.8	43.7±1	41.1±1	35±4	**29**±3	38.7±3	43.4±3.6	46.4±3	33.5±6	39±6	**28**±5	56±7
IID:(80, 10)	60	**70**±0.1	67±0.5	67.5±0.4	66±2	**15**±0.7	17±0.7	21.6±3.6	23.4±2	10.7±1	11±2	**8**±2	27±7
ResNet101	80	36.2±1	**38**±0.7	21.3±3	33±3.5	**24**±3	35.2±3	69.7±5	57±5	40±10	29±4	**21**±9	68±9
IID:(68,15)	60	**52.4**±1	51±0.8	45.7±2	48±3	**19**±1	20.6±3	38.2±4.5	32±6	23±4	22±11	**14.5**±1	47±7
		***CIFAR-100 with* 20 *superclasses* (L = 20)**											
MobileNet	80	36±0.7	36±0.4	27.5±0.3	**38**±0.4	31±1.7	**29**±1.6	32±3	31.1±2	17±2	16±1	**17**±3	38±2
IID:(44,28)													
ResNet101	80	23.6±1	20±1.7	**26**±2.4	23±1.6	42±4	**40.4**±6	54.1±2.2	45.6±5	39±6	42±7	**25**±5	60±8
IID:(38,29)													
		***CIFAR-100 with all classes* (L = 100)**											
MobileNet	80	27±0.1	27±0.1	28±0.2	**28**±0.1	48.6±1	49±0.8	**46**±1	46.7±2	13±2	13±2	**11**±1	22±0.5
IID:(33, 42)													
		SVHN											
MobileNet	80	77.4±0.4	77.7±0.4	**78**±0.5	77±0.6	24±0.3	24±0.4	**22**±0.2	24±0.7	**6**±1	7.7±1	7±0.5	9±1
IID:(89,5)													
ResNet101	80	63±2	**67**±3	66.2±1.6	59.7±2	**40**±2	46±3	42±1	54.9±1	7.7±4	**6**±3	8±5	11±6
IID:(88,6.2)													

Table 1: Global accuracy, Performance Gap, and Backward Transfer (reported median of the last five rounds across all runs) for cyclic & random order (ϕ is the scale of the clients in the cyclic order). The parenthesis under the model type shows the global accuracy and the PG for IID data. The upward/downward arrows indicate that large and small values are desirable, respectively.

one after the other. In other words, ϕ gives the scale of repetition of clients of the same type. In the random order, while such repetitions may occur, they may vary from round to round, and thus, ϕ is meaningful only in the cyclic order.

Table 1 shows the global accuracy, PG, and BW for the random and cyclic order with different values of ϕ in a variety of models and datasets. We observe that, in most of the lines, *the best accuracy, PG, and BW are achieved by the cyclic order (for any ϕ)*, with a few corner cases where random has slightly better accuracy than cyclic, but higher PG. Also, we observe that among the different values of ϕ, lower values yield the best accuracy, indicating that diversity in the training sequence is crucial. This finding aligns with other studies in the literature on CL, e.g., (Lin et al. 2023), but has not been showcased in SFL, until now. When ϕ is large, the server consecutively processes ϕ clients' data that have the same dominant label. This leads to more frequent updates related to a single label, allowing the model to learn this label better, but at the cost of forgetting previous ones.[1] Even though training seems to be more stable (i.e., lower BW), this leads to an increased PG, which implies higher disparity among labels' performance (leading to intraCF).

Both the PG and the BW metrics show that structured processing order policies (at the server), such as the cyclic order, lead to lower forgetting and better training stability when compared to unstructured ones, e.g., random order. This aligns with existing theoretical results on CL and linear regression (Evron et al. 2022). Moreover, as we can already observe in Table 1 (in terms of accuracy and PG), many characteristics of forgetting in CL also occur in SFL (e.g., that larger and more complex models are more sensitive to CF (Ramasesh, Lewkowycz, and Dyer 2021)). Hence, despite the differences in the two settings (CL and SFL), forgetting in these settings carries similarities, which can help to better understand CF and intraCF in SFL.

[1] A small exception is observed in SVHN, which appears to be less sensitive to ϕ because its label distribution is uneven; thus, labels with more samples dominate model updates regardless of ϕ.

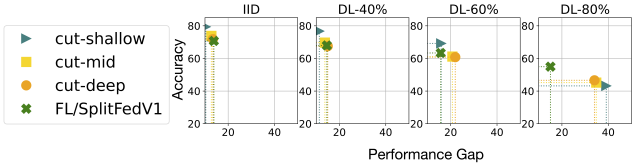

Figure 3: Accuracy (y-axis) and PG (x-axis) achieved by MobileNet/CIFAR-10 for SFL with different cuts (4, 15, and 23, resp.) and cyclic order ($\phi = 10$), and FL/SplitFedV1.

Cut Layer and IntraCF. Fig. 3 depicts the global accuracy and the PG for different cut layers (in SFL), and for SplitFedv1 (Thapa et al. 2022), a variant of SL. In SplitFedv1, part-1 is trained as in SFL, and part-2 is trained at the server using a copy for each client, while all copies are aggregated at the end of each round. It has been shown that its performance is the same as that of FL (Gao et al. 2021; Thapa et al. 2022), and hence, we also denote it by FL. For high data heterogeneity, i.e., 80%-DL, as we go from shallow to deep cut layer, the accuracy increases and the performance gap drops. We recall that the deeper the cut layer is, the larger part-1 becomes, i.e., more layers are trained at the clients (in parallel) and aggregated at the end of the round. Hence, *a deep cut layer highlights the effect of CF in part-1, while a shallow cut layer highlights the effect of intraCF on CF.* The former is related to the aggregation under data heterogeneity, while the latter to the processing order

at the server. These results imply that the impact of CF in part-1 (model drifting) is smaller than the effect of CF in part-2 (intraCF), i.e., *the cut layer balances the impact of model drifting of part-1 and intraCF of part-2*.

Due to the previous observation, we expect that the deeper the cut layer is in SFL, the more similar the model performs to FL. We see that, in terms of accuracy, SFL performs better than FL under IID data. As data heterogeneity increases, however, FL performs better (54% vs 43%, under 80%-DL), as also observed by (Gao et al. 2020), as a result of aggregating the entire model. In terms of CF, under 80%-DL, there is a significant performance gap between FL and SFL, for any cut layer. However, we see something unexpected: the PG of SFL with shallow cut layer (which showcases intraCF) is very close to that of deep cut layer (which showcases the CF of part-1). This means that although FL performs better under high heterogeneity, implying that aggregation has a positive impact on CF, the fact that *the last layers of the model are processed at the server (sequentially) leads to high CF in SFL*. The importance of the last layers of the model to the training and CF has also been shown in CL in (Ramasesh, Dyer, and Raghu 2021; Zilly et al. 2021).

4 Hydra: A Novel Method to Mitigate CF

We observed that intraCF contributes more to the phenomenon of forgetting in SFL than the model drifting of part-1, with the last layers playing an important role. Methods in related work concerning CL or FL often utilize multiple blocks of higher layers, where each one of them is allocated to a different task, e.g., (Gurbuz and Dovrolis 2022; Kim, Liu, and Ke 2022; Chen et al. 2023; Hemati et al. 2025). In such methods, often called *multi-head*, the lower layers are shared among all tasks, while there exist multiple higher layers. We notice that the design of SFL naturally aligns with the multi-head architecture, as the higher layers are handled only by the server. Leveraging this together with the benefit of averaging under heterogeneous data (as shown by the accuracy and PG of FL in Fig. 3), we propose the *first multi-head solution for SFL*, named *Hydra*, to tackle CF in SFL. Hydra derives from our insights in Sec. 3.

Design Details. Hydra extends SFL by modifying its workflow at the level of the server. Fig. 4 presents Hydra's design and workflow. In detail, part-1 is trained in the same way as before (solely handled by the clients), while part-2 (managed by the server) is split and composed of: *(i)* a unique part-2a, shared among all clients and updated sequentially, and *(ii)* G multiple versions of part-2b (heads), which are updated in parallel and aggregated (through `FedAvg`) at the end of each round. Each head corresponds to a label or a superclass (i.e., multiple labels). Further, the data corresponding to a group of clients is assigned to a specific head based on the highly represented labels in their data. This is done through a *mapping* that the server performs, where it assigns the different outputs (activations) of part-2a to the different heads during forward propagation. A natural choice for the number of heads is the number of labels, i.e., $G = L$, following the multi-head approaches in CL. However, G can be tuned according to the server's memory capacity to reduce overhead, as we discuss below.

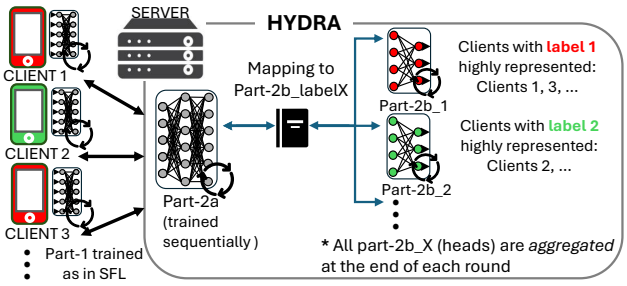

Figure 4: The workflow of Hydra, the proposed method to mitigate catastrophic forgetting in SFL.

The group of clients corresponding to each head is defined at the beginning of the training to cluster clients with similar data distributions, reducing large gradients that could overwrite previous updates. Homogeneity within a group ensures minimal weight changes, effectively addressing CF. This grouping requires each client to send information on the distribution of labels in its local data (as a vector of length L) before the training starts. The *grouping algorithm* is based on resource allocation problems and aims at grouping clients with similar label distributions while ensuring that the repartition of clients among the groups is balanced across all groups. In detail, it assigns clients to groups in a greedy way: for each group, the algorithm finds the client with the largest number of samples from the corresponding label and repeats this process until all clients have been assigned to exactly one group. Finally, by iterating through the groups instead of the clients, the repartition of clients into the groups is balanced. Note that the same mapping is then used inversely by the server during backward propagation.

Unlike the existing multi-head approaches in CL, whose final trained model contains multiple heads, e.g., (Kim, Liu, and Ke 2022), Hydra's final model contains a single head derived from aggregating all the different heads, making multi-head more suitable for SFL. In fact, our evaluation (Sec. 5) shows that aggregated heads excel in performance when compared to typical multi-head architectures. To the best of our knowledge, Hydra is the first attempt to fine-tune the multi-head technique for SFL. Also, other important design decisions of Hydra include the processing order of clients' activations through part-2a and the length of heads (part-2b). To this end, we thoroughly study the impact of these design choices in Sec. 5. Finally, we note that Hydra differentiates from the SL variant USplit (Vepakomma et al. 2018), in which the model is split into 3 parts, and the last (3rd) model part is processed at the clients and aggregated (at the end of each round). Indeed, in Hydra, the number G of heads is bounded by L, making it less prone to model drifting w.r.t. USplit, where C model parts are aggregated.

Hydra's Overhead. In terms of computing cost, there is a one-time $O(G \cdot C)$ cost for running the grouping algorithm at the beginning of training to assign clients into the different heads. During the training of part-2, there is no additional overhead since only one head is updated per batch update. Regarding the aggregation of heads at the end of every round, this can be implemented locally at the server and run

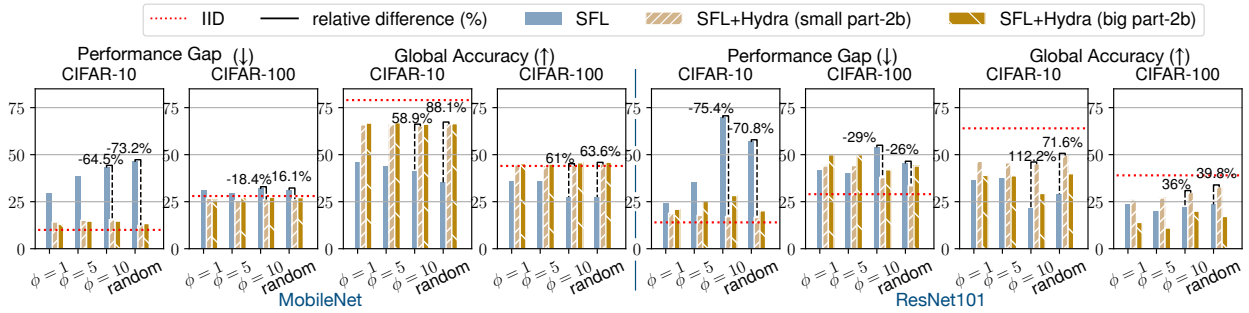

Figure 5: Performance gap and global accuracy (median) in SFL with and without Hydra with 80%-DL. Over-the-bar text shows the percentage of decrease/increase of the PG and global accuracy, respectively.

Dataset	Policy	Global Accuracy ↑ Sharding	Dirichlet ($\alpha = 0.3$)	Dirichlet ($\alpha = 0.1$)	Performance Gap ↓ Sharding	Dirichlet ($\alpha = 0.3$)	Dirichlet ($\alpha = 0.1$)
CIFAR10	SFL+Hydra	**72 ±0.4**	**59 ±0.4**	**49 ±0.2**	**10 ±0.4**	**21 ±1**	**32 ±0.8**
IID:(80, 10)	SFL	50 ±0.4	50 ±0.5	40 ±0.6	36 ±1	43.7 ±0.5	45.5 ±1
TinyImageNet	SFL+Hydra	**34 ±0.1**	**45 ±0.1**	**41 ±0.3**	**49 ±0.6**	**44 ±0.6**	**44.5 ±1**
IID:(50, 40)	SFL	27 ±0.3	**45 ±0.3**	31 ±1	62 ±1.6	46 ±1	63 ±0.7

Table 2: Global accuracy and performance gap achieved by MobileNet with $\phi = 10$ under different data partitions.

in parallel with the aggregation of part-1. We stress that Hydra induces no computational overhead on the clients' side, since it only concerns part-2 of the model (at the server).

In terms of memory overhead, this depends mainly on the number G and the length of the heads, but also on the implementation specifics. In practice, when the heads consist of only the last 2 layers, *the memory overhead can be as little as* $G \cdot 6$MB, or $G \cdot 10$MB when the heads consist of the last 6 layers. This estimation concerns Mobilenet and CIFAR-10 with a batch size of 64, and is based on the memory needed for storing the model weights and the intermediate activations/gradients. In fact, our evaluation in Sec. 5 encourages small heads since Hydra performs better when compared to the case with larger heads. Similarly, we show that Hydra leads to reduced CF when compared to vanilla SFL, even with a small G. Finally, Hydra induces a negligible communication overhead at the beginning of the training where each client sends a vector containing the information on its label distribution (that is needed for the mapping algorithm).

5 Experiments

In this section, we thoroughly evaluate the performance of the proposed method Hydra. Fig. 5 presents the results from SFL with and without Hydra (denoted by SFL+Hydra and SFL, respectively). These results concern multiple aspects: the processing order of part-2a at the server, datasets, ML models, and the selection of the heads' length.

Processing Order of Part-2a. We first focus on the case where Hydra's heads are small, and, in particular, their length is only 2 layers. This means that the second cut layer (i.e., the last layer of part-2a) is located at layer 24 and 33 for MobileNet and ResNet101, respectively. We notice that Hydra (SFL+Hydra) under **cyclic order** (with any ϕ) significantly outperforms the baseline (SFL). In detail, by employing Hydra, the labels' performance gap decreases by up to 64.5% and 75.4% for MobileNet and ResNet101 on CIFAR-10, respectively, while, at the same time, the global accuracy increases by 58.9% and 112.2%, respectively, approaching the accuracy under IID data. Furthermore, Hydra has a robust performance regardless of the client scale (ϕ), unlike the baseline SFL. From the results for the **random order**, we observe that Hydra reduces PG by up to 73%, thus providing a stable training even under a random order.

Length of Heads (Part-2b) in Hydra. We focus now (in Fig. 5) on Hydra with small and large lengths of heads (e.g., for the large heads, the second cut layer is at layers 20 and 30 for MobileNet and ResNet101, respectively). In general, for MobileNet, the two cases perform similarly, i.e, Hydra with small heads achieves slightly better PG, while Hydra with larger heads has slightly better accuracy, indicating that MobileNet is not affected by the heads' length. However, ResNet101, a more complex model, seems to be more sensitive to the length of heads since Hydra with small heads always outperforms the case of larger heads, in terms of both accuracy and PG, a side effect of model drifting. Therefore, a small length of heads seems to be the best design choice not only in terms of memory overhead (as discussed in Sec. 4), but also in terms of performance.

Types of Data Partitioning. In Sec. 4, Hydra was defined for any data partitioning. While Fig. 5 presented evaluation results of Hydra for the DL partition, Table 2 presents results in three additional types of data partitioning, namely Dirichlet with $\alpha = 0.1$ and 0.3 (where lower α indicates higher heterogeneity), and Sharding method with two dominant labels. The results demonstrate that *Hydra improves over the baseline (SFL) in all the tested data partitioning methods*.

Policy	Gl. Acc. ↑	PG ↓	BW ↓
SFL+Hydra	**66.5±0.1**	**13.4±1**	**9±0.9**
SplitFedV1/FL	55±0.2	15.7±1.5	14±2.7
SplitFedV3	43±0.3	22.6±0.8	20±2.5
MultiHead	47±0.3	36.6±0.9	25±2.6
MergeSFL	65±0.5	21±1.8	13±1.3
SplitNN	53±1	44±1	11±0.3

Table 3: Comparison of Hydra with state-of-the-art methods in MobileNet with CIFAR-10 and 80%-DL.

Policy	N° heads	Gl. Accuracy ↑ cyclic	Gl. Accuracy ↑ random	Perf. Gap ↓ cyclic	Perf. Gap ↓ random
SFL	N/A	41.1±1	35±4	43.4±3.6	46.4±3
SFL+Hydra	10	**65**±0.1	**66.5**±0.1	**15.4**±0.6	**13.4**±1
SFL+Hydra	5	56±1	57±1	19.3±1	23.1±1.5
SFL+Hydra	2	49±1	47±1	32±2	34±2

Table 4: Ablation study on the number of Hydra's heads (G) with MobileNet, CIFAR-10, 80%-DL, and IID:(80, 10).

Comparison with State-of-the-Art Methods. In Table 3, we evaluate Hydra (i.e., SFL+Hydra) using a random processing order (i.e., capturing a more generic scenario) and MobileNet. We compare it with other policies, i.e., SplitFedV1/FL (Thapa et al. 2022), SplitFedV3 (Madaan, Kulkarni, and Pant 2022), Multihead (Chen et al. 2023), MergeSFL (Liao et al. 2024), and SplitNN (Vepakomma et al. 2018). All of these methods have different workflows than SFL. Table 3 shows that Hydra significantly outperforms the state-of-the-art. Specifically, methods in the literature fail to improve on all metrics (indicating an unstable training), while *Hydra consistently excels in all metrics (accuracy, PG, and BW)*. Importantly, Hydra outperforms the other multi-head approaches (i.e., SplitFedV3 and MultiHead). This highlights that Hydra's aggregation of the heads is a key feature over the classic multi-head approaches. Moreover, MergeSFL achieves a global accuracy comparable to the one of Hydra. However, a significant drawback of MergeSFL lies in its implementation, which requires synchronization among clients and more frequent updates (due to micro-batches). Finally, among all the methods in the literature, SplitNN is the only one where the server processes clients' data in a specific order (typically random). As a result, SplitNN has the worst (i.e., highest) performance gap among all methods due to forgetting.

Closing the Labels' Performance Gap. To better understand Hydra's performance, we dive into the per-position accuracy (i.e., the accuracy of the labels that are dominant in clients that are processed at a certain position, see Sec. 3.1). In Fig. 6, we see that Hydra effectively reduces the labels' performance gap and stabilizes the training, even in a highly heterogeneous scenario, i.e., 80%-DL. In detail, unlike the baseline (SFL), there is no significant gap among the different positions. We note that the peaks and troughs in the subplot on the right are a result of the processing order under non-IID data. Finally, the global accuracy improves over SFL, i.e., 44% instead of 28%, after 100 rounds.

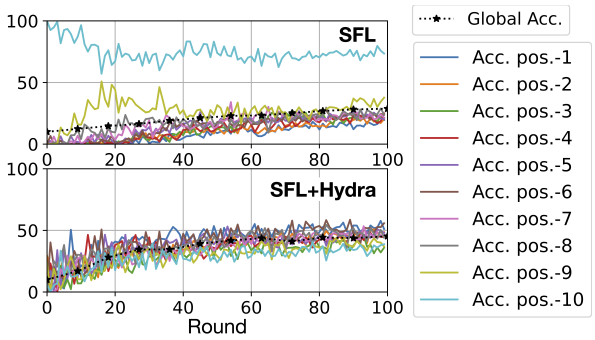

Figure 6: Per-position accuracy in SFL, and SFL+Hydra for ResNet101 with CIFAR-10 and 80%-DL partition.

Ablation Study on the Number of Heads G. As discussed in Sec. 4, Hydra's memory overhead can be adjusted by reducing the length of part-2b and/or G. For the former, we showed that a smaller part-2b performs better. Table 4 reports the performance of Hydra for different values of G for CIFAR-10 and different processing orders of part-2a (cyclic with $\phi = 10$ and random). For CIFAR-10, we define G superclasses based on the semantic relationships among labels, similar to (Bai et al. 2021). For example, when $G = 2$, we consider the two superclasses: animals and no-animals. We see that Hydra performs best when $G = L$, but still outperforms the baseline SFL in all metrics, even for a smaller number of heads. These results highlight an interesting trade-off between memory overhead and performance.

6 Conclusion and Discussion

In this work, we studied the phenomenon of catastrophic forgetting as a result of data heterogeneity in SFL. Our empirical analysis was performed in a variety of scenarios (e.g., ML models, datasets, etc.) allowing us to identify key parameters and derive insights into CF in SFL. Based on these, we proposed Hydra, a novel mitigation method. We showed that it successfully alleviates the forgetting while increasing the global accuracy and closing the labels' performance gaps. Hydra induces minimal computing and memory overhead (that can be further reduced through design choices) and outperforms baselines and state-of-the-art methods.

Limitations and Future Work. Despite the absence of theoretical claims in this work, our empirical analysis of CF in SFL paves the way towards a theoretical/convergence analysis of SFL with respect to key parameters such as the cut layer and the processing order at the server. Such an analysis could also be the stepping stone towards establishing theoretical guarantees for Hydra. Furthermore, an interesting direction for future work is to incorporate client selection policies into the analysis of CF, but also in Hydra, similar to related work in FL, e.g., (Lee et al. 2022). Finally, our ablation study on the number of Hydra's heads revealed that semantic relationships among labels may play an important role in CF and in SFL's training in general. While this aspect was not the primary focus of this work, we believe that it should be studied in the future, as it has been done for continual learning (Ramasesh, Dyer, and Raghu 2021).

Acknowledgments

This work was funded by the Horizon MSCA Postdoctoral Fellowship OPALS (grant agreement 101210495), the Spanish Ministry of Economic Affairs and Digital Transformation, the European Union-NextGenerationEU through the projects 6G-RIEMANN (TSI-063000-2021-147) and MAP-6G (TSI-063000-2021-63), and the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme through the CONFIDENTIAL6G EU project (Grant Agreement No 101096435). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

Bai, Y.; Yan, X.; Jiang, Y.; Xia, S.-T.; and Wang, Y. 2021. Clustering effect of adversarial robust models. *Advances in Neural Information Processing Systems*, 34: 29590–29601.

Chen, C.; Kevin, I.; Wang, K.; Li, P.; and Sakurai, K. 2023. Flexibility and Privacy: A Multi-Head Federated Continual Learning Framework for Dynamic Edge Environments. In *2023 Eleventh International Symposium on Computing and Networking (CANDAR)*, 1–10. IEEE.

Evron, I.; Moroshko, E.; Ward, R.; Srebro, N.; and Soudry, D. 2022. How catastrophic can catastrophic forgetting be in linear regression? In *Conference on Learning Theory*, 4028–4079. PMLR.

Feng, X.; Jia, R.; Luo, C.; Leung, V. C.; and Xu, W. 2024. SLwF: A Split Learning Without Forgetting Framework for Internet of Things. *IEEE Internet of Things Journal*.

Feng, X.; Luo, C.; Chen, J.; Huang, Y.; Zhang, J.; Xu, W.; Li, J.; and Leung, V. C. 2023. IoTSL: Toward Efficient Distributed Learning for Resource-Constrained Internet of Things. *IEEE Internet of Things Journal*, 10(11): 9892–9905.

Gao, Y.; Kim, M.; Abuadbba, S.; Kim, Y.; Thapa, C.; Kim, K.; Camtep, S. A.; Kim, H.; and Nepal, S. 2020. End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things. In *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 91–100. IEEE Computer Society.

Gao, Y.; Kim, M.; Thapa, C.; Abuadbba, A.; Zhang, Z.; Camtepe, S.; Kim, H.; and Nepal, S. 2021. Evaluation and optimization of distributed machine learning techniques for internet of things. *IEEE Transactions on Computers*, 71(10): 2538–2552.

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Gurbuz, M. B.; and Dovrolis, C. 2022. NISPA: Neuro-Inspired Stability-Plasticity Adaptation for Continual Learning in Sparse Networks. In *International Conference on Machine Learning*, 8157–8174. PMLR.

Hafi, H.; Brik, B.; Frangoudis, P. A.; Ksentini, A.; and Bagaa, M. 2024. Split federated learning for 6G enabled-networks: Requirements, challenges and future directions. *IEEE Access*, 12: 9890–9930.

Han, P.; Huang, C.; Tian, G.; Tang, M.; and Liu, X. 2024. Convergence Analysis of Split Federated Learning on Heterogeneous Data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Hemati, H.; Pellegrini, L.; Duan, X.; Zhao, Z.; Xia, F.; Masana, M.; Tscheschner, B.; Veas, E.; Zheng, Y.; Zhao, S.; et al. 2025. Continual learning in the presence of repetition. *Neural Networks*, 183: 106920.

Howard, A.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. sl, sn. *arXiv preprint arXiv:1704.04861*.

Hu, M.; Cao, Y.; Li, A.; Li, Z.; Liu, C.; Li, T.; Chen, M.; and Liu, Y. 2024. FedMut: Generalized federated learning via stochastic mutation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 12528–12537.

Jimenez G., D. M.; Solans, D.; Heikkila, M.; Vitaletti, A.; Kourtellis, N.; Anagnostopoulos, A.; Chatzigiannakis, I.; et al. 2024. Non-IID data in Federated Learning: A Systematic Review with Taxonomy, Metrics, Methods, Frameworks and Future Directions. *arXiv preprint arXiv:2411.12377*.

Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T.; Mars, J.; and Tang, L. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1): 615–629.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Kim, G.; Liu, B.; and Ke, Z. 2022. A multi-head model for continual learning via out-of-distribution replay. In *Conference on Lifelong Learning Agents*, 548–563. PMLR.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

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

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

Lee, G.; Jeong, M.; Shin, Y.; Bae, S.; and Yun, S.-Y. 2022. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems*, 35: 38461–38474.

Leys, C.; Ley, C.; Klein, O.; Bernard, P.; and Licata, L. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4): 764–766.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, Y.; Li, Q.; Wang, H.; Li, R.; Zhong, W.; and Zhang, G. 2024. Towards efficient replay in federated incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12820–12829.

Li, Y.; and Lyu, X. 2023. Convergence analysis of sequential federated learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 36: 56700–56755.

Liao, Y.; Xu, Y.; Xu, H.; Wang, L.; Yao, Z.; and Qiao, C. 2024. Mergesfl: Split federated learning with feature merging and batch size regulation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2054–2067. IEEE.

Lin, S.; Ju, P.; Liang, Y.; and Shroff, N. 2023. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, 21078–21100. PMLR.

Ma, Y.; Xie, Z.; Wang, J.; Chen, K.; and Shou, L. 2022. Continual Federated Learning Based on Knowledge Distillation. In *IJCAI*, 2182–2188.

Madaan, H.; Kulkarni, M. G. V.; and Pant, A. 2022. Vulnerability due to training order in split learning. In *ICT Systems and Sustainability: Proceedings of ICT4SD 2021, Volume 1*, 103–112. Springer.

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

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Y.; et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 4. Granada.

Qi, D.; Zhao, H.; and Li, S. 2023. Better Generative Replay for Continual Federated Learning. In *The Eleventh International Conference on Learning Representations*.

Ramasesh, V. V.; Dyer, E.; and Raghu, M. 2021. Anatomy of Catastrophic Forgetting: Hidden Representations and Task Semantics. In *International Conference on Learning Representations*.

Ramasesh, V. V.; Lewkowycz, A.; and Dyer, E. 2021. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.

Swartworth, W.; Needell, D.; Ward, R.; Kong, M.; and Jeong, H. 2023. Nearly optimal bounds for cyclic forgetting. *Advances in Neural Information Processing Systems*, 36: 68197–68206.

Thapa, C.; Arachchige, P. C. M.; Camtepe, S.; and Sun, L. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8485–8493.

Tirana, J.; Tsigkari, D.; Iosifidis, G.; and Chatzopoulos, D. 2024. Workflow Optimization for Parallel Split Learning. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 1331–1340.

Vepakomma, P.; Gupta, O.; Swedish, T.; and Raskar, R. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33: 7611–7623.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2024. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xia, Z.; Hu, M.; Yan, D.; Liu, R.; Li, A.; Xie, X.; and Chen, M. 2025. MultiSFL: Towards Accurate Split Federated Learning via Multi-Model Aggregation and Knowledge Replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 914–922.

Xu, C.; Hong, Z.; Huang, M.; and Jiang, T. 2022. Acceleration of Federated Learning with Alleviated Forgetting in Local Training. In *International Conference on Learning Representations*.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zilly, J.; Achille, A.; Censi, A.; and Frazzoli, E. 2021. On plasticity, invariance, and mutually frozen weights in sequential task learning. *Advances in Neural Information Processing Systems*, 34: 12386–12399.