Heterogeneous SplitFed: Federated Learning with Trainable and Untrainable Clients

Juliana N D da Silva, Stefan Duffner, Virginie Fresse

Heterogeneous SplitFed: Federated Learning with Trainable and Untrainable Clients

Juliana N. D. da Silva*†, Stefan Duffner*, Virginie Fresse†
* INSA Lyon, CNRS, Universite Claude Bernard Lyon 1, LIRIS, UMR5205, 69621 Villeurbanne, France
{juliana.damurie, stefan.duffner}@insa-lyon.fr
†Hubert Curien Laboratory, UMR CNRS 5516, 42000 Saint-Etienne, France
{virginie.fresse}@univ-st-etienne.fr

***Abstract*—With the advent of edge computing and distributed learning paradigms, the integration of low-resource devices and embedded systems within these frameworks has become a focal point of numerous research initiatives. These resource constraints, particularly in memory and computational capacity, are exacerbated by the demands of increasingly complex neural network models that are deployed on such devices. SplitFed Learning (SFL) has been proposed as an innovative approach that combines two prominent distributed machine learning strategies, namely federated learning (FL) and split learning (SL), which facilitates model usage among clients with resource limitations while preserving their privacy. However, there are specific devices where training is difficult or impossible, which SFL, FL, or SL do not consider. For instance, devices based on Field Programmable Gate Arrays (FPGAs) may face such challenges. Despite this, these devices could still benefit from the federation and contribute to it with their own data. Therefore, in this paper, we introduce a new federated learning approach for deep neural networks, called Heterogeneous SplitFed Learning (HSFL), designed to support low-resource clients that are only capable of performing model inference and that can cope with heterogeneous data, thus enabling their active participation. This enhances privacy while improving model performance in collaboration with clients owning more substantial computational resources. We demonstrate empirically on image classification benchmarks and common deep learning models that HSFL can match the performance of other FL approaches that accommodate heterogeneous data, maintaining efficacy and including clients with limited resources.**

***Index Terms*—Federated Learning, heterogeneous devices, heterogeneous data, edge computing**

I. Introduction

As the number of interconnected devices continues to grow, collecting user data through increasingly varied methods, there is a corresponding rise in legislative measures. Notable examples include the General Data Protection Regulation (GDPR) [1] and California Privacy Rights Act (CPRA) [2], designed to safeguard users against the unauthorized dissemination of their personal information. In response to these challenges, edge computing and distributed machine learning have been advanced as potential solutions [3, 4]. These technologies promise to enhance user proximity to computational resources, thereby delivering expedited, low-latency services that prioritize privacy [5].

Federated Learning (FL) [4, 6, 7, 8] aims to decentralize machine learning processes from the cloud infrastructure, enabling client-side training while preserving data privacy. In this approach, a shared model is maintained in the cloud, which is continuously updated and improved based on the weights of each client's model. Subsequently, the refined joint model is redistributed to the clients for further enhancement.

For clients with resource limitations who still need to train a complex model in the cloud without sharing their private data, Split Learning (SL) emerges as a viable solution [9]. This method divides the neural network model into two parts: one belonging to the client and the other to the server. The client trains on its private data and then sends its "smashed data" and labels to the server. The server then trains its part of the model and returns the activations and training results to the client, enabling the client to perform the backpropagation step and update its model. However, SL doesn't allow you to combine the models of several clients, so you can't take advantage of all the data that may be distributed across several locations.

To this end, Thapa et al. [10] proposed SplitFed Learning (SFL), a method that combines Split Learning (SL) and Federated Learning (FL). This approach enables clients with resource constraints to train models without sharing their data, addressing the issue of time overhead inherent in SL. Additionally, SFL introduces the approach of model aggregation like in standard FL, further enhancing its effectiveness. Despite the solution proposed by Thapa et al. [10], we can still face issues with devices limited by memory constraints, or those incapable of training models and solely performing model inference. These devices may include smart sensors, FPGAs, or small embedded systems.

Within FL research, one topic of study concerns settings with non independently and identically distributed (IID) data, where clients may have different data classes and varying amounts of data. This non-IID data reflects the heterogeneity in real-life scenarios, where clients possess different and unbalanced datasets. Incorporating such data into FL remains an open challenge [11]. Another challenge regarding heterogeneity is the varying resources among clients, such as different computational capacities and memory limitations.

Some solutions, such as model pruning, gradient compression, and other techniques, can reduce the cost of running a neural network model on resource-constrained devices with a minor reduction in accuracy. However, none of these solutions consider devices that lack the capability to train a model while

still maintaining privacy when participating in FL.

Our contributions can be summarized as follows:

- We propose a new federated learning approach that enables the participation of clients that have private data but do not have the necessary computational resources and capacities to perform the local training by effectively collaborating with other clients that have these capabilities.
- Our approach is flexible and not only integrates clients of different capacities and resource limits but also copes well with non-IID and extremely heterogeneous data distributions as we will show experimentally.

In this paper, we demonstrate a proof of concept and focus on the algorithmic aspects evaluating the classification performance of different variants of our approach, showing that the inclusion of untrainable clients with their data is beneficial for the entire federation. However, we will not cover any practical implementations on hardware and specific resources-limited devices.

II. Background and Related Work

The foundation of Federated Learning (FL) lies in the FedAvg algorithm, introduced in the seminal paper on FL [6]. In this algorithm, a global model is initialized and distributed to all participating clients. Each client then receives the model and performs local training for a predefined number of epochs. Subsequently, the locally updated model weights are sent back to the global server. Aggregation occurs by averaging the set of model parameters from all clients, resulting in a new set of global model parameters. Communication costs are reduced by allowing clients to perform multiple epochs of local training before transmitting their parameters. However, in heterogeneous environments the performance may deteriorate the global performance, as the local iterations can cause the local optimal model to diverge from the global optimal model.

To address these challenges, alternatives such as FedProx [12] and FedDyn [13] have been developed. These algorithms introduce a regularization term during the model loss calculation, which accounts for the distance between the global and local model parameters. This penalty on large discrepancies helps mitigate divergence, allowing clients with heterogeneous data and computational capabilities to perform varying numbers of local updates without significantly deviating from the global model. Nevertheless, this approach still requires that each client can train the complete model.

The SplitFed Federated Learning framework (version 1) SFLV1 [10] offers a solution for conducting FL with resource-limited clients by splitting the model into two parts: a client model and a server model. On the server, models are executed in parallel and subsequently aggregated using FL algorithms like FedAvg. This approach results in a globally optimized model based on contributions from all clients, while allowing resource-limited clients to perform backpropagation and optimize their local models.

Among the works addressing resource heterogeneity and limitations, several studies focus on model pruning, model rescaling, and distillation as strategies to adapt models for clients with varying capabilities. These techniques aim to maintain model performance while accommodating the diverse resource constraints of participating clients.

For example, HeteroFL [14] is a model rescaling method for enabling heterogeneous clients to train local models with different levels of complexity. Clients can use a subset of the global model's parameters, facilitating more efficient and tailored training processes. HeteroFL also addresses the limitations of traditional Batch Normalization [15] by introducing Static Batch Normalization (sBN), which has shown superior performance in producing consistent and improved results. However, clients with limited resources are constrained to having local models with fewer parameters, leading to lower performance.

Another approach called FedDF [16] proposes a knowledge distillation framework to cope with heterogeneous environments and models. The algorithm uses unlabeled data in the distillation process, which can be generated by Generative Adversarial Networks (GAN), and model fusion. The performance of the framework depends on the effectiveness of the synthetic data and the distillation process.

ScaleFL [17] tries to address these problems by combining rescaling strategies and knowledge distillation, also focusing on clients that are heterogeneous in terms of resources and data. To adapt the neural network model on clients, it uses the 'scale down deep neural network' mechanism, which adjusts the model in depth and width, as well as knowledge distillation during the aggregation process. One of the disadvantages of ScaleFL is the complexity of the aggregation algorithm, which can make the effective combination of local models a significant challenge, especially in environments with a wide diversity of resources among clients.

Although these existing approaches can effectively incorporate heterogeneous clients with limited resources to some extend, none of them is capable of including clients that are not able to train at all and that can only do inference (e.g. FPGA-based devices, smart sensors or other low-resource IoT devices). Our proposed federated learning approach enables the effective collaboration of trainable and untrainable clients such that the data from all clients benefits the overall federation without compromising the privacy of each individual client.

III. Heterogenous SplitFed

A. System Architecture and Data

We consider the problem where we aim to implement federated learning among K heterogeneous clients. Among these K clients, we can identify J clients with resource limitations, who are unable to perform the training of the neural network, i.e. gradient computation, backpropagation and weight update. These clients are referred to as *untrainable clients*. Additionally, within this set of K clients, there are L clients who have the capacity to train a complete model; these are termed *trainable clients*. Thus, we have $K = J \bigcup L$.

Each client k has a local dataset $\mathcal{D}^k$ that remains fixed during training. Therefore, we consider the total dataset to

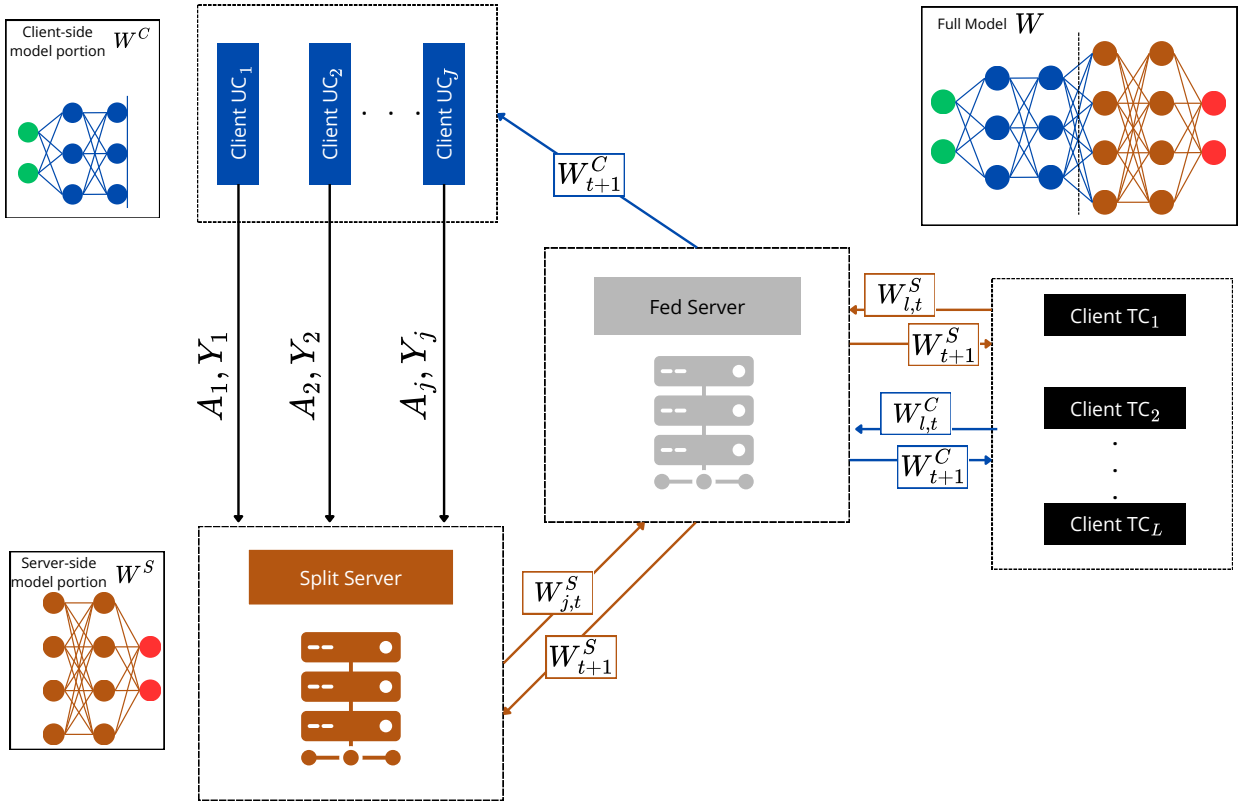

Fig. 1. Overview of the Heterogeneous SplitFed Learning (HSFL) approach. In the solution, we have Trainable Clients (TC) represented in black on the right side, which possess the complete model W. On the left side, in blue, are the Untrainable Clients (UC) who face certain limitations; they only have a portion of the model. These clients perform only inference using their input data. Afterward, the final part of the model is trained on the split server, and the set of parameters from the models trained on the split server is sent to the FedServer to be aggregated with the parameter sets from the TCs. Once the FedServer completes the aggregation, it returns the set of parameters to each client and also to the split server, which maintains separate copies of each client's model.

be represented by $\mathcal{D}$, where $\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}^k$. The dataset $\mathcal{D}$ contains a total of C classes, with an example from this dataset denoted by $X = (x, y)$.

As in SFL, there is a Fed Server and a Split Server (called Main Server in SFL). However their role and tasks are slightly different in our approach. Figure 1 illustrates the overall system architecture and training procedure. The *Fed Server* is collecting the models from all clients, trainable and untrainable, combines them according to the chosen aggregation algorithm (here FedAvg) and sends the combined model back to the corresponding devices. Whereas the *Split Server* is responsible for doing the inference and training of the server-side model (i.e. last layers) of the untrainable clients and communicates these models with the Fed Server. The training procedure is detailed in the next section.

This setting is different from the SFL approach since :

- A subset of clients performs only inference on the client-side model (i.e., the first layers) without engaging in training;
- The trainable clients contain the full model;
- Model aggregation is performed entirely on the Fed Server.

These distinctions enable our approach to effectively learn a deep neural network model in a federated and privacy-preserving manner with a heterogeneous set of devices both in terms of computational and memory resources and in terms of client data distributions.

B. Training

As in traditional SL, the full model is divided into two parts: the client part (W^C), i.e. the first layers up to the "cut layer" (green and blue layers in Fig. 1), and the server part (W^S), i.e. the last layers of the neural network (brown and red layers in Fig. 1).

The *untrainable clients* S_t^{TC} at round t use only the client part of the model W^C and perform only inference on their side. These clients send the output of their model, the "smashed data" A_j and labels Y_j to the Split Server, which will train the server part of the model W_j^S ($j \in S_t^{UC}$). After completing a training round for all untrainable clients, the Split Server sends its local weights $W_{j,t}^S$ to the Fed Server.

In parallel, the *trainable clients* S_t^T perform training on the full model, and after a training round, they send their local weights of the complete model (i.e. all layers) to the Fed Server. The Fed Server then aggregates the two models – the client and server model – and returns the global model weights to the respective clients and server. The complete view of the

approach is summarized in Figure 1, and the training algorithm is described more formally in Algorithms 1, 2 and 3.

Privacy is ensured on all clients. On trainable clients, local data is never transmitted. On untrainable clients, we follow the principle of SFL where only activations of intermediate layers ("smashed data") are transmitted. In our case, we use deep convolutional neural network, and smashed data are feature map activations of relatively "late" layers that make reconstruction of the input (images) very difficult.

Algorithm 1: Heterogeneous SplitFed with Trainable and Untrainable Clients

Notations: S_t^{TC}: set of L Trainable Clients at time t,
S_t^{UC}: set of J Untrainable Clients at time t,
S_t: set of K Trainable + Untrainable Clients at time t

Fed Server **executes:**
- Initialize W_0^C, W_0^S (global client and server-side model);
- Send W_0^C to all K clients ;
- Send W_0^S to L Trainable Clients and Split Server;
- **for** *round t= 0,1,...* **do**
  - **forall** *l in* S_t^{TC} ***(in parallel)*** **do**
    - $W_{l,t}^C, W_{l,t}^S \leftarrow$ ProcessTrainableClient() ;
  - **forall** *j in* S_t^{UC} ***(in parallel)*** **do**
    - $W_{j,t}^S \leftarrow$ ProcessUntrainableClient() ;
  - $m_l \leftarrow \sum_{l \in S_t^{UC}} n_l$;
  - $W_{t+1}^C, \leftarrow \sum_{l \in S_t^{UC}} \frac{n_l}{m_l} W_{l,t}^C$ *// aggregate* W^C ;
  - $m_k \leftarrow \sum_{k \in S_t} n_k$;
  - $W_{t+1}^S, \leftarrow \sum_{k \in S_t} \frac{n_k}{m_k} W_{k,t}^S$ *// aggregate* W^S;
  - Send W_{t+1}^C, W_{t+1}^S to all K clients ;

Split Server **executes:**
- Retrieve updated models $W_{j,t}^S$ from FedServer ;
- **while** *client data batches to process* **do**
  - Retrieve A_j, Y_j from client j ;
  - $\hat{Y}_j \leftarrow$ ForwardPropagationServer();
  - $\mathcal{L} = \mathcal{L}(\hat{Y}_j, Y_j)$;
  - *// Compute gradients and back-propagate*
  - $\nabla \mathcal{L} \leftarrow$ LossGradients() ;
  - $W_{j,t}^S \leftarrow W_{j,t}^S - \eta \cdot \nabla \mathcal{L}$;
- Send $W_{j,t}^S$ to Fed Server $\quad (j \in S_t^{UC})$;

IV. Experiments

Datasets. We conducted experiments in image classification using the FEMNIST [18], CIFAR-10, and CIFAR-100 [19] datasets. We utilized the standard training and testing splits as detailed in Table I. To enhance model robustness, we applied data augmentation techniques, specifically implementing a random horizontal flip with a probability of 0.5 and random cropping, following [17].

Algorithm 2: ProcessTrainableClient()

Retrieve model updates $W_{l,t}^C, W_{l,t}^S$ from FedServer ;
$\mathcal{B} \leftarrow$ (split $\mathcal{D}_l$ into batches of size B) ;
for $e \leftarrow 1..E$ **do**
- **for** *batch* $b \in \mathcal{B}$ **do**
  - *// Perform forward propagation*
  - $\hat{Y}_l \leftarrow$ ForwardPropagationTrainableClient();
  - $\mathcal{L} = \mathcal{L}(\hat{Y}_l, Y_l)$;
  - *// Compute gradients and back-propagate*
  - $\nabla \mathcal{L} \leftarrow$ LossGradients() ;
  - *//* $W_{l,t} = (W_{l,t}^C, W_{l,t}^S)$
  - $W_{l,t} \leftarrow W_{l,t} - \eta \cdot \nabla \mathcal{L}$;

Send updated $W_{l,t}^C, W_{l,t}^S$ to Fed Server ;

Algorithm 3: ProcessUntrainableClient()

Retrieve model updates $W_{j,t}^C$ from FedServer ;
$\mathcal{B} \leftarrow$ (split $\mathcal{D}_j$ into batches of size B) ;
for $e \leftarrow 1..E$ **do**
- **for** *batch* $b \in .\mathcal{B}$ **do**
  - *// Perform forward propagation on client*
  - *// to compute smashed data*
  - $A_j \leftarrow$ ForwardPropagationUntrainableClient();
  - Send A_j, Y_j to Split Server ;

Dataset	Train Size	Test Size	# Classes	Resolution
FEMNIST	37K	5K	62	28x28
CIFAR-10	50K	10K	10	32x32
CIFAR-100	50K	10K	100	32x32

TABLE I
STATISTICS OF THE DATASETS USED IN IMAGE CLASSIFICATION EXPERIMENTS.

Data heterogenity. In our experiments, all clients participated in the aggregation during each round. To simulate data heterogeneity, we divided the data according to a Dirichlet distribution with a concentration parameter α. We utilized three levels of non-IID distribution: $\alpha = 0.1$, $\alpha = 1$, and $\alpha = 10$. In the Dirichlet distribution, an α value closer to zero represents extreme heterogeneity, while higher values indicate a more balanced distribution. With $\alpha = 0.1$ some classes may be missing (i.e. have 0 examples) for some clients.

The FEMNIST dataset is inherently a federated learning (FL) dataset, meaning it is originally divided by clients.

	FEMNIST	CIFAR-10	CIFAR-100
Centralized Learning (CL)	87.02	92.27	64.04
Federated Learning - 4 clients	89.27 ± 0.82	$70,93 \pm 0.99$	51.22 ± 0.11
Federated Learning - 6 clients	80.15 ± 0.41	73.76 ± 0.26	41.04 ± 0.48

TABLE II
BASELINE RESULTS FOR CENTRALIZED LEARNING AND CLASSICAL FEDERATED LEARNING (FEDAVG) WITH ALL TRAINING DATA FROM THE THREE DATASETS.

(TC,UC)	FEMNIST Accuracy (%) for Different α Values			
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	IID
$(2,0)^a$	95.1 ± 0.6	86.8 ± 1.9	79.2 ± 5.4	84.3 ± 0.1
$(2,2)^a$	94.4 ± 2.1	87.4 ± 2.5	84.7 ± 1.6	86.1 ± 0.1
$(2,0)^b$	90.5 ± 5.5	86.2 ± 0.8	82.6 ± 1.7	79.1 ± 3.3
$(2,4)^b$	94.0 ± 2.9	87.0 ± 3.3	83.6 ± 1.3	82.3 ± 1.1
$(4,0)^c$	93.7 ± 2.6	86.8 ± 4.9	84.5 ± 1.3	78.6 ± 1.0
$(4,2)^c$	94.0 ± 1.4	87.3 ± 3.2	84.3 ± 1.1	80.6 ± 2.5
$(4,0)^d$	93.7 ± 2.9	85.5 ± 0.8	82.6 ± 3.1	74.6 ± 0.8
$(4,4)^d$	92.3 ± 3.7	84.4 ± 4.6	81.5 ± 4.8	77.6 ± 1.9

TABLE III
AVERAGE ACCURACY (%) WITH STANDARD DEVIATION FOR RESNET-34 ON THE FEMNIST DATASET FOR VARIOUS CONFIGURATIONS. THE FIRST COLUMN SHOWS THE DATA SPLITS BETWEEN TC AND UC FOR A) 4, B) 6, C) 6, AND D) 8 CLIENTS IN TOTAL. WHEN UC IS 0, THIS REPRESENTS A TRADITIONAL FEDERATED LEARNING (FL) SCENARIO WITH ONLY THE TC DATA PROPORTION.

(TC,UC)	CIFAR-10 Accuracy (%) for Different α Values			
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	IID
$(2,0)^a$	91.3 ± 3.1	84.8 ± 5.7	82.6 ± 2.7	78.2 ± 0.5
$(2,2)^a$	91.3 ± 1.9	88.1 ± 4.7	81.6 ± 1.8	79.3 ± 1.6
$(2,0)^b$	94.4 ± 0.8	84.6 ± 2.5	74.7 ± 3.6	74.6 ± 0.7
$(2,4)^b$	90.7 ± 8.0	83.1 ± 3.9	81.0 ± 1.6	75.1 ± 2.3
$(4,0)^c$	90.9 ± 4.6	88.6 ± 4.3	80.9 ± 4.7	74.9 ± 2.9
$(4,2)^c$	92.5 ± 3.8	87.6 ± 4.7	81.8 ± 2.5	74.6 ± 1.9
$(4,0)^d$	96.8 ± 2.4	93.6 ± 2.7	80.1 ± 2.7	73.0 ± 2.9
$(4,4)^d$	92.2 ± 5.1	86.5 ± 3.8	79.2 ± 5.7	71.2 ± 1.9

TABLE IV
AVERAGE ACCURACY (%) WITH STANDARD DEVIATION FOR RESNET-34 ON CIFAR-10 DATASET.

(TC,UC)	CIFAR-100 Accuracy (%) for Different α Values			
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	IID
$(2,0)^a$	77.9 ± 2.7	64.2 ± 0.7	50.3 ± 0.4	48.3 ± 0.6
$(2,2)^a$	74.6 ± 5.5	61.8 ± 6.2	52.9 ± 0.7	48.1 ± 0.2
$(2,0)^b$	74.3 ± 5.0	55.2 ± 1.7	41.6 ± 0.0	38.2 ± 0.5
$(2,4)^b$	71.9 ± 4.9	56.0 ± 4.9	42.7 ± 1.5	38.1 ± 0.3
$(4,0)^c$	76.4 ± 1.8	57.0 ± 3.0	43.9 ± 0.5	37.4 ± 0.3
$(4,2)^c$	74.0 ± 3.2	58.1 ± 3.5	44.5 ± 1.4	37.8 ± 0.4
$(4,0)^d$	77.1 ± 4.1	53.3 ± 6.9	31.4 ± 4.6	21.2 ± 1.8
$(4,4)^d$	72.8 ± 5.5	54.2 ± 6.1	33.7 ± 5.0	23.8 ± 1.8

TABLE V
AVERAGE ACCURACY (%) WITH STANDARD DEVIATION FOR RESNET-34 ON CIFAR-100 DATASET.

However, in our experiments, we combined the data from some clients and redistributed it according to a Dirichlet distribution.

When dividing the data according to the Dirichlet distribution, we created four configurations: (a) 4 parts divided among 2 TC and 2 UC clients, (b) 6 parts among 2 TC and 4 UC clients, (c) 6 parts among 4 TC and 2 UC clients, and (d) 8 parts among 4 TC and 4 UC clients. For experiments without UC clients, we used the same data division but excluded UC clients, meaning their data was not used in training.

Experimental Setup. We conducted our experiments using the ResNet-34 architecture, a widely adopted model in federated learning studies. The model was split between clients and the server, where the first 7 layers operated on the client side and the remaining layers on the server. Each experiment consisted of 200 communication rounds with a single epoch per round.

We compared our models in terms of classification performance, i.e. we computed the average (sample-wise) accuracy (micro average) on the test sets of the three different datasets which is distributed among the clients.

Optimization and Training. We employed Stochastic Gradient Descent (SGD) with a learning rate of 0.1 for the FEMNIST dataset, and 0.1 for both the CIFAR-10 and CIFAR-100 datasets. The FedAvg algorithm was employed to aggregate the models across clients.

Results. For comparison, Table II shows the average accuracies of ideal baseline methods and scenarios, i.e. centralised learning (CL) and federated learning (FedAvg) on all training data with IID splits for 4 and for 6 clients.

Table III shows the average accuracies on the FEMNIST test set for varying number of trainable and untrainable clients and with splitting the data into a) 4, b) 6, c) 6, and d) 8 parts. In general, we observe that in the presented examples, more heterogeneous cases exhibit better performance than homogeneous ones. This can be attributed to the fact that heterogeneous clients have more examples within fewer classes, allowing them to somehow specialize in those classes. Thus, although models are aggregated after each round with one epoch, the parameters of the specialist networks are effectively utilized and combined. The literature also provides other examples where heterogeneous cases demonstrate superior performance, as seen in studies such as [20, 21].

Compared to the ideal baselines of Table II, we can see that the cases with $\alpha = 1$ have results close to the CL IID case and for the cases with $\alpha = 0.1$ we have better average results than the CL IID case. For more balanced settings or larger number of clients, our approach is slightly below the ideal baselines.

When comparing successive lines in Table III, i.e. cases a), b), c, and d), it can be seen that adding untrainable clients is beneficial to the overall performance and to the federation. This is especially the case for IID data and for $\alpha > 1$. For more heterogeneous data distributions, the fact of adding untrainable clients does not harm the performance and at least maintains the overall accuracy.

Tables IV and V show the results on the CIFAR-10 and

CIFAR-100 datasets respectively. Similar trends to FEMNIST can be observed. That is, the models attain a higher accuracy in more heterogeneous settings.

V. Conclusion

In this paper, we presented a novel federated learning approach, called HSFL, that enables the participation in training of clients that have very limited hardware resources and are not capable of training a model but can do inference. Our approach is an extension of the SplitFed method that allows for such a scenario. We experimentally showed on three different image classification benchmarks that HSFL can effectivly learn with a combination of trainable and untrainable clients where the latter contribute positively to the entire federation. We further experimented with different levels of heterogeneity of the data splits between clients and show that our approach is robust to these data distribution shifts.

In future work, we will investigate further the resource requirements in terms of network communication, do further model optimisations like quantisation and knowledge distillation and also add model personalization to our method.

References

[1] J. P. Albrecht, "How the gdpr will change the world," *Eur. Data Prot. L. Rev.*, vol. 2, p. 287, 2016.

[2] P. Bukaty, *The California Privacy Rights Act (CPRA)–An implementation and compliance guide*. IT Governance Ltd, 2021.

[3] G. Bao and P. Guo, "Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges," *Journal of Cloud Computing*, vol. 11, no. 1, p. 94, 2022.

[4] J. Wu, F. Dong, H. Leung, Z. Zhu, J. Zhou, and S. Drew, "Topology-aware federated learning in edge computing: A comprehensive survey," *ACM Computing Surveys*, 2023.

[5] P. Li, G. Cheng, X. Huang, J. Kang, R. Yu, Y. Wu, and M. Pan, "Anycostfl: Efficient on-demand federated learning over heterogeneous edge devices," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

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

[7] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[8] P. Kairouz, B. H. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021. [Online]. Available: https://inria.hal.science/hal-02406503

[9] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[10] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8485–8493.

[11] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," 2021.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[13] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=B7v4QMR6Z9w

[14] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," *CoRR*, vol. abs/2010.01264, 2020. [Online]. Available: https://arxiv.org/abs/2010.01264

[15] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated Learning on Non-IID Features via Local Batch Normalization," Sep. 2020. [Online]. Available: https://openreview.net/forum?id=6YEQUn0QICG

[16] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *CoRR*, vol. abs/2006.07242, 2020. [Online]. Available: https://arxiv.org/abs/2006.07242

[17] F. Ilhan, G. Su, and L. Liu, "ScaleFL: Resource-adaptive federated learning with heterogeneous clients," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 24 532–24 541.

[18] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *CoRR*, vol. abs/1812.01097, 2018. [Online]. Available: http://arxiv.org/abs/1812.01097

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009, https://www.cs.toronto.edu/ kriz/cifar.html.

[20] Y. Shi, Y. Zhang, Z. Huang, X. Yang, L. Shen, W. Chen, and X. Wang, "Heterogeneous federated learning with splited language model," 2024. [Online]. Available: https://arxiv.org/abs/2403.16050

[21] R. Ye, Z. Ni, F. Wu, S. Chen, and Y. Wang, "Personalized federated learning with inferred collaboration graphs," in *International Conference on Machine Learning*. PMLR, 2023, pp. 39 801–39 817.