Hierarchical Federated Transfer Learning: A Multi-Cluster Approach on the Computing Continuum

Sabtain Ahmad
Vienna University of Technology
Vienna, Austria
sabtain.ahmad@tuwien.ac.at

Atakan Aral
University of Vienna, Vienna, Austria
Umeå University, Umeå, Sweden
atakan.aral@umu.se

***Abstract*—Federated Learning (FL) involves training models over a set of geographically distributed users. We address the problem where a single global model is not enough to meet the needs of geographically distributed heterogeneous clients. This setup captures settings where different groups of users have their own objectives however, users based on geographical location or task similarity, can be grouped together and by inter-cluster knowledge they can leverage the strength in numbers and better generalization in order to perform more efficient FL. We introduce a Hierarchical Multi-Cluster Computing Continuum for Federated Learning Personalization (HC3FL) to cluster similar clients and train one edge model per cluster. HC3FL incorporates federated transfer learning to enhance the performance of edge models by leveraging a global model that captures collective knowledge from all edge models. Furthermore, we introduce dynamic clustering based on task similarity to handle client drift and to dynamically recluster mobile (non-stationary) clients. We evaluate the HC3FL approach through extensive experiments on real-world datasets. The results demonstrate that our approach effectively improves the performance of edge models compared to traditional FL approaches.**

***Index Terms*—federated transfer learning, hierarchical collaborative learning, dynamic clustering**

I. Introduction

Federated learning (FL) is a privacy-preserving distributed on-device learning framework that employs privately available data on client devices such as tablets, smartphones, and IoT devices. In traditional machine learning, the client is supposed to send data to the cloud server for training. However, sending data to the cloud server raises privacy concerns and may not even be possible due to regulations such as GDPR. FL relaxes this restriction by allowing clients to locally train models and aggregate them over the cloud parameter server. The objective is to compute a single global model in a collaborative and privacy-preserving manner [1]. The training routine comprises the following basic steps: (1) the parameter server broadcasts the initial global model to all participating clients, (2) each client learns a local model using its private data, (3) the parameter server collects and aggregates the local models to compute a new global model. The updated global model is sent back to the clients for another training round.

In FL, it is typically assumed that clients can be grouped into a single cluster, and a single global model can be learned to generalize over clients' individual learning tasks [2]. The attempt to collaboratively train a model over a diverse set of clients with different storage and computation capacities suffers from data and system heterogeneity [3], [4]. Exploiting data heterogeneity and task similarity is particularly crucial for applications such as environmental monitoring, recommender systems, and smart cities. For instance, sensors deployed on different geographical locations (tributaries) over a river to monitor its health may obtain different measurements as the characteristics of the river change over distance due to elevation, climate, and geology [5]. This indicates that leveraging the data heterogeneity among different sensors and geographical locations is of potential interest.

We propose a hierarchical, multi-cluster approach to personalization in FL, inspired by the prevalence of hierarchical models in social sciences. Data often exhibit hierarchical structures in various social science fields, where entities are nested within larger communities or clusters, acknowledging that entities within the same cluster may be more similar to each other than to those in other clusters [6]. Our approach leverages the known hierarchical structure and enables us to simultaneously learn three models: (1) a global model that captures trends and patterns for all clients, (2) a cluster-specific model that captures shared characteristics among clients within a cluster, and (3) a personalized model that captures unique features of each individual client. More specifically, the devices on the lower level of the computing continuum train device-specific personalized models and are grouped into clusters based on intrinsic similarity. Each cluster is assigned to an edge node responsible for training a cluster-specific model, while the cloud server is responsible for generating the global model by aggregating only the updates from edge nodes. Additionally, we introduce transfer learning to fine-tune the edge models using the global model to allow inter-cluster knowledge sharing while retaining the cluster-specific features.

By utilizing the hierarchical structure, our approach can capture both shared and unique information among clients, resulting in better personalization. Additionally, the cluster-specific models enable us to capture the cluster-level characteristics that are not captured in the global model, which can lead to better performance. Furthermore, the global model can be used as a starting point for cluster-specific and personalized models, reducing the overall training time.

Our work makes several key contributions. First, we propose a clustering scheme to group geographically dispersed clients based on task similarity and data distribution. Second, we introduce a hierarchical multi-cluster approach that enables the simultaneous learning of three models: a global model, a cluster-specific model, and a personalized local model. This approach leverages the computing continuum to organize devices into a hierarchy based on their computing capabilities, improving scalability and reducing the burden on the cloud server. Third, instead of overriding the edge models in each iteration, we employ transfer learning to encourage inter-cluster knowledge sharing. We empirically evaluate the performance of our proposed algorithm, called Hierarchical Multi-Cluster Computing Continuum for Federated Learning Personalization (HC3FL), in both IID, non-IID, and dynamic clustering settings where client identities are unknown and may change over time. Our experiments demonstrate that HC3FL outperforms existing state-of-the-art methods, such as FedAvg [1] and HierFL [7], for all three settings. The rest of the paper is organized as follows. Related work is discussed in Sec. II. In Sec. III and IV, we introduce HC3FL and present experimental results. We conclude the paper in Sec. V.

II. Related Work

Ever since its inception, FedAvg [1] has become the preferred method due to its simplicity and effectiveness in most applications. However, the vanilla version suffers from concept drift, and its performance deteriorates in the case of non-IID [8]. This issue has been in the spotlight for a while, and numerous approaches have been proposed with a strong emphasis on personalization. For instance, MOCHA [2] employed multi-task learning to allow for personalization and to handle statistical heterogeneity. Another approach is to formulate the non-IID problem as a meta-learning problem with the objective of obtaining a single global model. Then, each client fine-tunes the model using its data [3], [9]. The underlying assumption is that the data distributions among heterogeneous clients are similar; therefore, the global model can serve as a suitable initialization.

Another reason for the inferior performance of the single global model in non-IID settings is the assumption that the diverse set of clients with individual preferences belongs to a single large set. This observation has forced researchers to steer their attention towards clustered FL, with some of the recent works focusing on clustering users based on data distribution or task similarity [10], [11]. IFCA [10] suggests leveraging data heterogeneity to improve accuracy and convergence speed by introducing a hierarchical clustering algorithm that groups clients based on data similarity. Similarly, in [11], the authors introduced a clustering algorithm based on privacy requirements, data similarity, and computational power.

The clustered FL approaches are effective yet heavy at the server; leading to high computation costs at the parameter server. Our work differs from the existing literature in several aspects. First, we cluster clients based on their data distribution, which allows us to form clusters based on the similarity of data characteristics among clients. Second, we introduce edge servers that act as intermediate nodes between the cloud server and clients and let us train one edge model per cluster. Specifically, all clients from a cluster send their updates to a specific node to train cluster-specific models. The distributed architecture reduces communication overhead and facilitates efficient model updates within each cluster. Third, we introduce a variation of HC3FL to handle dynamic clustering (HC3FL-DC) as the clients' associations may change over time due to concept drift. By allowing dynamic clustering, HC3FL-DC can adapt to shifting data patterns and maintain up-to-date cluster assignments for optimal personalization. Finally, we employ transfer learning to facilitate inter-cluster knowledge sharing and improve model performance. To the best of our knowledge, HC3FL is the first framework to introduce and integrate dynamic clustering and federating transfer learning in a hierarchical FL setting.

III. HC3FL

The motivation behind HC3FL is three-fold: (1) it is reasonable to assume that a small subset of devices among a large pool of users may share common interests or similar tasks. For instance, devices in closer proximity may contain similar information, but data characteristics may diverge for distant devices. By clustering clients with similar learning tasks, HC3FL aims to capture similar patterns and task similarity to improve the performance of individual models, (2) even clusters that are further apart and have heterogeneous data may still be able to contribute to improving the cluster-specific models, emphasizing the need for inter-cluster knowledge-sharing, (3) communication with the cloud server is often the most expensive step in FL, which can significantly be reduced by strategically placing edge nodes. By exploiting the hierarchical architecture and facilitating a significant portion of the communication within the edge nodes, HC3FL reduces the communication iterations with the cloud server, thereby minimizing the communication costs.

A. Problem Formulation

We consider a distributed learning setting where we have one parameter (cloud) server, m edge servers, and n clients. The clients can communicate with the edge servers, and the edge servers communicate with the parameter server through predefined communication channels. We assume there are $K = \{i : i = 1, ..., K\}$ data distributions, $D_1, ..., D_k$, and that n clients are partitioned into k disjoint clusters, $C_1, ..., C_k$ where each cluster c_i communicates only with a particular edge server e_j. We assume no knowledge about the cluster identity of each client. In addition, each client contains a local data set $D_n = \{x_l, y_l\}_{l=1}^{|D_n|}$ where x_l denotes the input sample and y_l is the corresponding label. In our problem, we aim

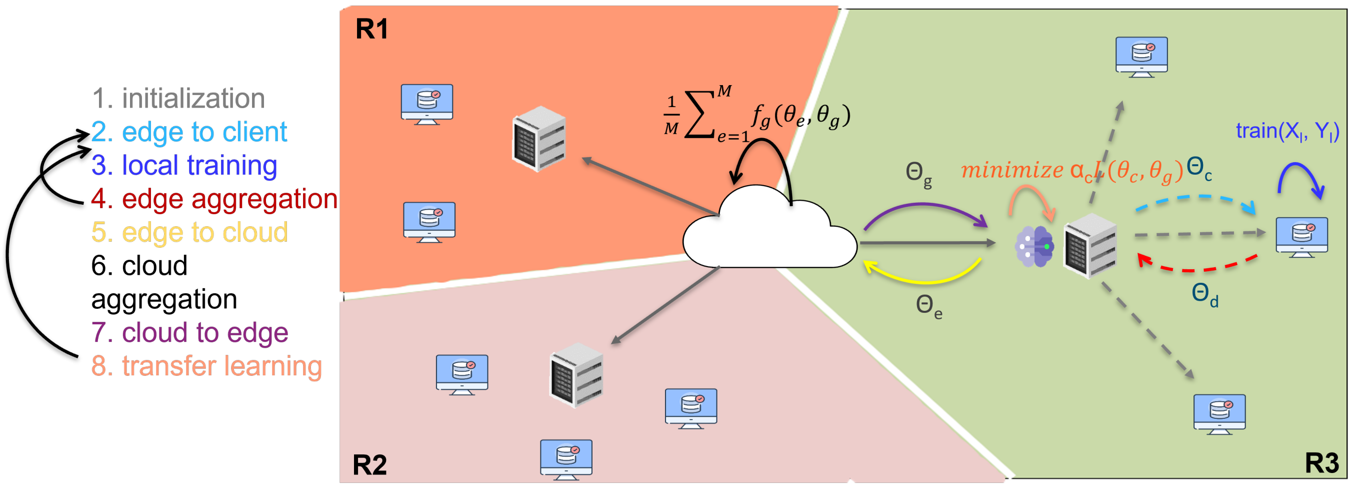


Fig. 1: An overview of the HC3FL framework.

to collaboratively train three models, one per entity (device, edge, and cloud), and enable inter-cluster knowledge transfer to train federated transfer models and improve performance.

B. Overview of the Framework

Our HC3FL architecture, depicted in Figure 1, utilizes geographically dispersed devices across three computing continuum levels to collaboratively train device, edge, and cloud models. The framework consists of four main procedures. First, we employ client-edge assignment to group similar clients together based on either geographical vicinity or similar data distributions. This ensures that clients within the same cluster have more similar models compared to clients in other clusters. Second, we implement hierarchical federated learning training. Initially, the clients' models, trained on their local data, are aggregated at the edge to compute cluster-specific edge models. These edge models then, in turn, are aggregated at the parameter server to generate the global model. Third, due to potential distribution divergence between clusters, we introduce transfer learning to exploit and incorporate knowledge from other clusters. Finally, in many federated learning scenarios, clients are assumed to be static, with their geographical locations remaining unchanged throughout the learning process. However, certain applications, such as autonomous driving and wearable health, involve mobile clients. Additionally, the client's distribution or the learning task itself may change over time due to concept drift, requiring a change in client edge/cluster associations. To cater to such scenarios, we employ dynamic clustering based on task similarity. This method allows for the re-clustering of mobile clients, ensuring efficient collaboration and model training. By incorporating these procedures into our HC3FL architecture, we achieve effective and adaptive FL in heterogeneous computing environments.

C. Hierarchical Federated Learning

We consider a hierarchical FL setup with one parameter server, m edge servers, and n clients. Within this setup, N_k clients are assigned to each edge server, where k represents the cluster index. Each client aims to optimize the model parameters $\mathbf{w_n}$ using its local training data. The standard loss function for a client n can be defined as;

$$M_n = \frac{1}{|D_n|}\sum_{l=1}^{|D_n|} f_n(\boldsymbol{x_l}, \boldsymbol{y_l}, \boldsymbol{w_n}) \tag{1}$$

where, f_n represents the client-specific loss function, $\boldsymbol{x_l}$ and $\boldsymbol{y_l}$ are the input and output data, respectively, and $|D_n|$ denotes the size of the local dataset for client n.

During the training process, each client n performs $\boldsymbol{t}$ number of local iterations and computes the local update at the t^{th} iteration;

$$\boldsymbol{w}_n^t = \boldsymbol{w}_n^{t-1} - \eta\nabla f_n(\boldsymbol{w}_n^{t-1}) \tag{2}$$

where η is the learning rate hyperparameter, and ∇f_n represents the gradient of the loss function wrt. model parameters.

After compiling the local updates, each client n within the cluster c_k transmits its local model $\boldsymbol{w}_n^t$ to its associated edge server e_k. The edge server then aggregates the models to compute a cluster-specific edge model $\boldsymbol{w_c}$ using the objective;

$$M_c = \frac{1}{N_k}\sum_{j=1}^{N_k} f_c(\theta_{j,c}, \boldsymbol{w}_c) \tag{3}$$

where $\boldsymbol{w}_c$ is the cluster model and $\theta_{j,c}$ is the local model of j^{th} client in cluster c .

At the edge aggregation step, after receiving the local updates, each edge node e_k performs local model aggregation to obtain the cluster-specific edge model $\boldsymbol{w_c}$ as follows;

$$\boldsymbol{w}_c = \frac{\sum_{n\in N_k}|D_n|\boldsymbol{w}_n^t}{|D_{N_k}|} \tag{4}$$

where $|D_{N_k}|$ is the aggregated data of all clients in cluster e_k

Following Equation 3, the global loss function is computed by aggregating the $\boldsymbol{w_c}$, resulting in the global model $\boldsymbol{w_g}$;

$$M_g = \frac{1}{K}\sum_{c=1}^{K} f_g(\boldsymbol{w}_c, \boldsymbol{w}_g) \tag{5}$$

where, f_g denotes the global loss function, and $\boldsymbol{w}_c$ and $\boldsymbol{w}_g$ represent the edge model of the e_k edge node and the global model, respectively.

To generate the global model, each edge node e_k uploads its edge model $\boldsymbol{w_c}$ to the parameter server. The parameter server then performs aggregation on the received edge models, resulting in the computation of the global model as follows;

$$\boldsymbol{w}_g = \frac{\sum_{c\in m} |D_{N_k}|\boldsymbol{w}_c}{|D|} \tag{6}$$

where D is the aggregated data set from all clients.

D. Federated Transfer Learning

The motivation for federated transfer learning stems from the observation that the edge models can learn the representations of the clients belonging to the same cluster fairly well. However, further apart clusters with slightly different distributions modeling the same task may still be able to capture some features that could be relevant and helpful for other edge models. As a result, the global model that aggregates all edge models can be assumed to encompass the characteristics of the entire set of edge models. Enabling knowledge transfer from the global model to the edge models can thus enhance the performance of the edge models and, consequently, improve the personal client models.

The aim of federated transfer learning is to optimize the performance of the edge models by incorporating knowledge from the global model. This can be achieved by minimizing the discrepancy between the edge and global models while preserving the individual features learned by each edge model. Therefore, the objective function is defined as;

$$\mathit{minimize} \sum_{c\in K} \alpha_c \cdot \mathcal{L}(\boldsymbol{w_c}, \boldsymbol{w_g}) \tag{7}$$

where $\mathcal{L}(\cdot)$ is a loss function that quantifies the discrepancy between two models, and α_c is a weight factor that determines the importance of each cluster's contribution to the objective.

For fine-tuning the global cloud model, the loss function can be defined as the difference in performance metrics between the two global and the edge models;

$$\mathcal{L}(\boldsymbol{w_c}, \boldsymbol{w_g}) = f(M_c) - f(M_g) \tag{8}$$

The objective function aims at minimizing the discrepancy between the cluster models and the global model while encouraging the edge models to acquire relevant knowledge from the global model while retaining their individual characteristics learned from the local data sets.

E. Federated Clustering

The aim of federated clustering is to partition the devices into clusters based on task similarity and ensure that the models trained on each local data set capture the characteristics of the respective clusters. The objective function is defined as;

$$\mathit{minimize} \sum_{c\in K}\sum_{z_{l,c}\in D_n} \mathcal{L}(z_{l,c}, \boldsymbol{w_c}) \tag{9}$$

where $Z_{l,c}$ is the data sample of n^{th} client of cluster c and $\mathcal{L}(\cdot)$ is a loss function that quantifies the dissimilarity between a data point and the edge models.

We use k-means and define the loss function as the squared Euclidean distance between data points and cluster centroid;

$$\mathcal{L}(z_{l,c}, \boldsymbol{w_c}) = ||z_{l,c} - \mathit{centroid}(c)||^2 \tag{10}$$

F. Algorithm

We propose an algorithm for hierarchical multi-cluster federated transfer learning (HC3FL). We discuss two variations of HC3FL, namely static clustering and dynamic clustering, presented in Algorithm 1. As the precursor to the main algorithm, a preprocessing step is introduced to create an initial set of clusters with similar clients and assign them to appropriate edge servers. HC3FL begins by initializing the global model, cluster-specific edge models, and clients' local models. The parameter server sends the global model to each edge server. Each client within the cluster associated with an edge server receives a copy as their local model. The number of global, edge, and local iterations are set as g, e, and t.

During each global iteration, the parameter server sends the global model to each edge server (step 1). Similarly, at each edge iteration, each edge server broadcasts the edge/cluster model to each client within the cluster (step 2). After receiving the model, each client trains the model using its local data for t local iterations and computes the local update (steps 3 & 4). After, clients send the local update to their respective edge servers (step 5). Each edge server performs edge aggregation similar to the weighted FedAvg [12] (step 6).

Each client's contribution to the edge model is relative to its local sample size. Each edge server sends the edge model back to clients, and steps 2-5 are repeated for $e-1$ steps. After e edge iterations, each edge server computes the cluster/edge model using the local updates from the clients associated with it and uploads the cluster model to the parameter server (step 7). After receiving the cluster models from each edge server, the parameter server performs the weighted aggregation to generate a global model update (step 8). Similar to edge aggregation, the contribution from each cluster to the global model is relative to the cluster sample size. The updated global model is then broadcast to all the edge servers.

Once the edge servers receive the updated global model, the algorithm proceeds to the transfer learning step (step 8), where the knowledge from the global model is shared with each cluster model. In the vanilla HC3FL approach, each edge server broadcasts its client model to the respective clients, and training continues until convergence or a specified number

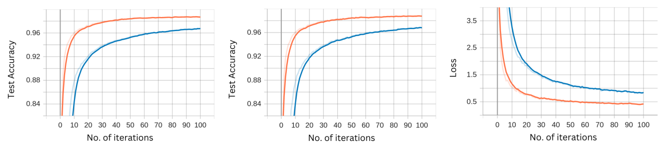
Algorithm 1 Hierarchical Multi-Cluster Federated Transfer Learning (HC3FL)

```
Input: client-edge mapping, local data sets $D_n$, # clusters $k$
Output: personalised client models $\boldsymbol{w_n}$, cluster-specific edge
        models $\boldsymbol{w_c}$, and a global cloud model $\boldsymbol{w_g}$
// Main HC3FL Loop for each global iteration do
    for each edge server $e_k$ do
        Broadcast global model $\boldsymbol{w}_g$ to $e_k$            // Step 1
        $\boldsymbol{w}_c \leftarrow$ federatedTransferLearning($\boldsymbol{w_g}, \boldsymbol{w_c}$)
        for each edge iteration do
            Broadcast cluster model $\boldsymbol{w}_c$ to all clients within $e_k$
             // Step 2
            for each client $n$ in cluster $k$ do
                Perform local training using $\boldsymbol{w}_n$ for $t$ local
                 iterations                                    // Step 3
                 Compute local update $\boldsymbol{w}_n^t = \boldsymbol{w}_n^{t-1} -$
                 $\eta \nabla M_n(\boldsymbol{w}_n^{t-1})$                // Step 4
                 Send local update to $e_k$                    // Step 5
            end
            Perform edge aggregation to compute $\boldsymbol{w}_c$
             // Step 6
        end
        Upload $\boldsymbol{w}_c$ to the parameter server          // Step 7
    end
    Perform global model aggregation to update $\boldsymbol{w}_g$
     // Step 8
end
// Transfer Learning                                         // Step 9
 Function federatedTransferLearning($\boldsymbol{w}_g$, $\boldsymbol{w}_c$):
                    // Perform TL using $\boldsymbol{w}_g$ and $\boldsymbol{w}_c$
    minimize $\alpha_c.\mathcal{L}(w_c, w_g)$
    return updated edge model $\boldsymbol{w}_c$
end

HC3FL-DC (Dynamic Clustering):;
 for each edge server $e_k$ in $m$ edge servers do
    Broadcast cluster model $\boldsymbol{w}_c$ to all participating clients
end
for each client $n$ do
    Estimate cluster identity $\hat{c} = \arg\min_{c\in[k]} F_n(\boldsymbol{w}_c)$
     if $n$ belongs to cluster $\hat{c}$ then
        perform steps 3 & 4
         Communicate the local update to the edge server
    end
end
```

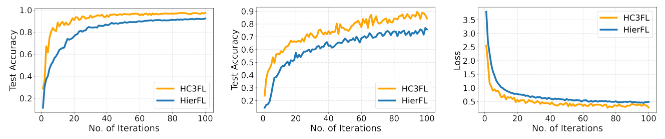
of global iterations, denoted as g. However, in HC3FL with dynamic clustering, the cluster models are broadcast to all participating clients. Utilizing the received cluster models and the local empirical loss F_i, each client estimates its cluster identity by finding the cluster model with the lowest loss, denoted as $\hat{c} = \arg\min_{c\in[k]} F_i(\theta_c^g)$. After estimating the cluster identities, clients with the same cluster identity perform local training (steps 3 and 4) and communicate their updates to their corresponding edge server.

IID Setting



(a) Global Model (b) Edge Model (c) Training Loss

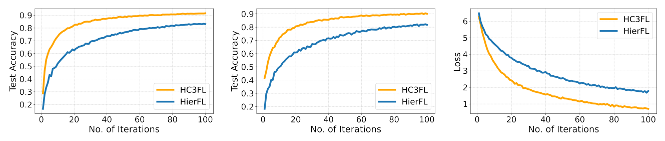
Non-IID Setting



(d) Global Model (e) Edge Model (f) Training Loss

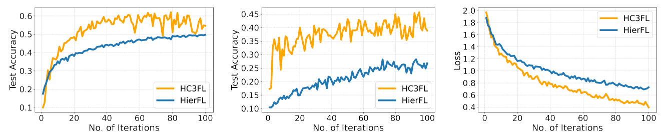
Fig. 2: MNIST Results

IID Setting



(a) Global Model (b) Edge Model (c) Training Loss

Non-IID Setting



(d) Global Model (e) Edge Model (f) Training Loss

Fig. 3: CIFAR Results

IV. Experiments

We evaluate the effectiveness of HC3FL in both IID and non-IID settings with dynamic clustering using MNIST [13] and CIFAR-10 [14] data sets. Both IID and non-IID configurations consist of 3 clusters, each containing 10 clients. We employ the non-IID data generation and distribution approach from [15] for simulating non-IID settings. We also generate clustered FL data sets based on these two data sets. In order to simulate a configuration where the data on different clients are generated from different distributions, we augment the data sets using rotation and generated rotated MNIST and CIFAR-10 data sets [10]. The rotated MNIST data set is split into 4 clusters by applying 0, 90, 180, and 270 degrees of rotation to the images. The data with the same rotation is divided into 10 clients resulting in 4 clusters in total, each of the size 10. We split the test data in the same fashion. Similarly, we create a rotated CIFAR-10 data set by creating two clusters with 0 and 180 rotations. For MNIST experiments, we use the fully connected neural network with a single hidden layer of size 200 and ReLU activations. For the CIFAR experiments, we use Resnet18 [16], and the data is preprocessed using standard data augmentation, such as random sampling and flipping.

TABLE I: Test accuracies (%) ± std on MNIST and CIFAR

	Clustered MNIST			Clustered CIFAR
m,n	2, 10	3, 10	4, 10	2, 25
HC3FL-DC	**94.63 ±0.03**	**95.20 ±0.03**	**97.44 ±0.02**	**83.05 ±1.37**
HierFL	89.54 ± 0.04	90.65 ± 0.08	92.37 ± 0.13	77.27 ± 0.39
FedAvg	66.32 ± 0.02	67.87 ± 0.04	69.05 ± 0.02	53.97 ± 1.19

A. IID and Non-IID Settings

We compare the performance of the proposed method with the HierFL [7] since it involves personalized client models, edge models, and a global cloud model. We evaluate and compare the performance of these two methods in both IID and non-IID settings for both data sets MNIST and CIFAR-10, as presented in Figures 2 and 3, respectively. The HC3FL outperforms HierFL; both the edge and global models achieve better performance than HierFL. On average, HC3FL achieves 3.12% and 9.6% increase in performance compared to HierFL in IID settings for MNIST and CIFAR, respectively. While in the case of non-IID, the HC3FL experiences 5.4% and 12.2% performance boost. The superior performance of the HC3FL is due to the fact that the HierFL randomly assigns clients to the edge servers, and aggregating clients with different learning tasks could result in inferior performance as reflected in Figures 2 and 3 (b & e). Furthermore, HC3FL benefits from the inter-cluster knowledge and retains the cluster-specific characteristics by fine-tuning the global model.

B. Dynamic Clustering

We compare our proposed HC3FL-DC with two baseline algorithms, i.e., FedAvg [1] and HierFL [7]. For MNIST experiments, the client participation is set to 1. For local client training, we chose $\tau = 10$ and learning rate $\gamma = 0.1$. For CIFAR, the participation rate was set to 0.5, the learning rate to 0.01, the learning rate decay to 0.995, and the batch size to 20. In the FedAvg (global model scheme), the algorithm does not consider cluster identities and tries to learn a single global model that can make predictions from all the distributions. While in HierFL, it introduces an edge layer between the clients and the cloud server, and clients are randomly assigned to the edge nodes.

The experimental results are shown in Table I. We can observe that the proposed approach HC3FL-DC performs better than both baselines. We observe that HC3FL-DC can gradually find the underlying cluster identities of the individual clients, and after the correct cluster is found, each client model is trained and tested using data with the same distribution, resulting in enhanced performance. The global model based on FedAvg performs worse as it tries to fit all the data from different distributions, and cannot provide personalized predictions.

V. Conclusion

In this paper, we proposed HC3FL, a hierarchical multi-cluster federated transfer learning approach, which incorporates a dynamic clustering algorithm (HC3FL-DC) to adaptively estimate the cluster identities of the clients to accommodate mobile and non-stationary data distributions. The experimental evaluations compared HC3FL with two baselines, HierFL, and FedAvg, on the widely used MNIST and CIFAR-10 datasets under both IID and non-IID settings with dynamic clustering. The empirical results demonstrate the superior performance of HC3FL over the baselines in all evaluated scenarios. In future, we will investigate the impact of the number of clusters on convergence and performance and the tradeoff between geo-proximity and task/data similarity in cluster formation.

Acknowledgment

The work presented in this paper has been supported by the CHIST-ERA grant CHIST-ERA-19-CES-005 and by the Austrian Science Fund (FWF): I 5201-N.

References

[1] 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.

[2] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[3] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[4] S. Ahmad and A. Aral, "Fedcd: Personalized federated learning via collaborative distillation," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2022, pp. 189–194.

[5] S. Ahmad, H. Uyanık, T. Ovatman, M. T. Sandıkkaya, V. D. Maio, I. Brandic, and A. Aral, "Sustainable environmental monitoring via energy and information efficient multi-node placement," *IEEE Internet of Things Journal*, 2023.

[6] S. Banerjee, A. Yurtsever, and M. H. Bhuyan, "Personalized multi-tier federated learning," in *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[7] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[8] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," *arXiv preprint arXiv:2002.04758*, 2020.

[9] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[10] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 586–19 597, 2020.

[11] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[12] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[13] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[14] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, no. 5, 2014.

[15] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

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