2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI)

CHESSFL: Clustering Hierarchical Embeddings for Semi-Supervised Federated Learning

Allen-Jasmin Farcas[1], Myungjin Lee[2], Ali Payani[2], Hugo Latapie[2], Ramana Rao Kompella[2], Radu Marculescu[1]
[1]*The University of Texas at Austin*, Austin, USA
Email: {allen.farcas, radum}@utexas.edu
[2]*Cisco Research*, San Francisco, USA
Email: {myungjinle, apayani, hlatapie, rkompell}@cisco.com

***Abstract*—Hierarchical Federated Learning (HFL) has shown great promise over the past few years, with significant improvements in communication efficiency and overall performance. However, current research for HFL predominantly centers on supervised learning. This focus becomes problematic when dealing with semi-supervised learning, particularly under non-IID scenarios. In order to address this gap, our paper critically assesses the performance of straightforward adaptations of current state-of-the-art semi-supervised FL (SSFL) techniques within the HFL framework. We also introduce a novel clustering mechanism for hierarchical embeddings to alleviate the challenges introduced by semi-supervised paradigms in a hierarchical setting. Our approach not only provides superior accuracy, but also converges up to 5.11× faster, while being robust to non-IID data distributions for multiple datasets with negligible communication overhead.[1]**

***Index Terms*—Hierarchical Federated Learning, Semi-Supervised Learning, Edge Devices, Data Heterogeneity, Data Privacy, Communication Efficiency, Internet-of-Things**

I. Introduction: Setting the Board

The Opening: Federated Learning (FL) has emerged as a transformative paradigm in machine learning (ML), enabling distributed training across decentralized devices, while keeping data private. McMahan et al. [1] first introduce FL with the concept of communication rounds representing the interplay between the clients and the cloud. First, the cloud broadcasts the global model to a subset of clients among all available clients (*e.g.*, 10% of all the clients). Clients, typically edge devices, begin training on their local data and send their updated local models back to the cloud. Finally, the cloud aggregates all local models and obtains a new global model. This entire process repeats until convergence, typically for several hundred communication rounds.

As the number of Internet-of-Things (IoT) devices increases annually, with nearly two billion more devices in 2023 than in 2022 [2], the range of these devices spans from powerful edge servers to less capable edge devices and IoT gadgets. While FL focuses on decentralization and data privacy, it can become inefficient when considering heterogeneous devices spread across various geographical areas with variable network connectivity. Hierarchical Federated Learning (HFL) emerged to overcome these very challenges. By introducing a client-edge-cloud hierarchical system architecture, HFL [3] trains k_1 local epochs on the client-side data and executes k_2 edge aggregations at the edge servers among connected clients before aggregating the edge models in the cloud to obtain a new global model. This hierarchical approach reduces the communication cost since edge servers are physically closer to the clients and enhances the computational efficiency of clients, thus allowing for quicker local model updates and edge aggregations at the edge servers. A hierarchical decentralized system is thus more scalable compared to classical FL. Besides scalability, client's privacy can be enhanced by the client-edge-cloud hierarchy provided by HFL [4]. On top of the resulting benefits of decentralization through hierarchy, when compared to local differential privacy [5], hierarchical differential privacy [6] highlights an opportunity for privacy strengthening due to differential privacy noise being injected multiple times throughout the hierarchy, at the intermediary edge servers and at the cloud server.

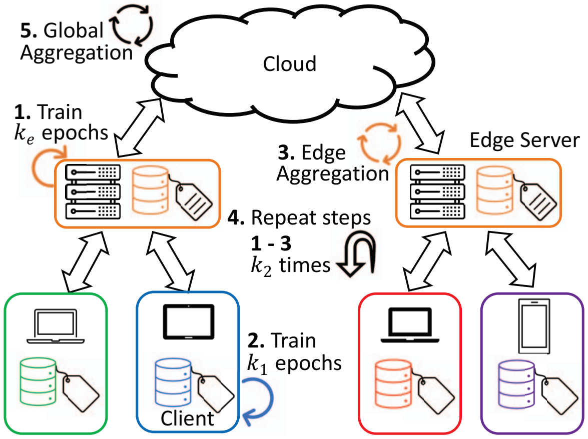

Fig. 1: Our proposed unified Semi-Supervised Hierarchical Federated Learning (SSHFL) evaluation framework. Edge servers (shown with orange border) have *labeled* data, while heterogeneous devices (shown with different colors, green, blue, red, purple) have their own local *unlabeled* data. Similar to classic HFL, we run steps 1 through 5 and evaluate the global model's test accuracy after a given number of communication rounds.

Queen's Gambit: Fully supervised learning in FL is quite unrealistic [7]. First, the collection and labeling of data is

[1] Code is available at: https://github.com/SLDGroup/CHESSFL

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00015

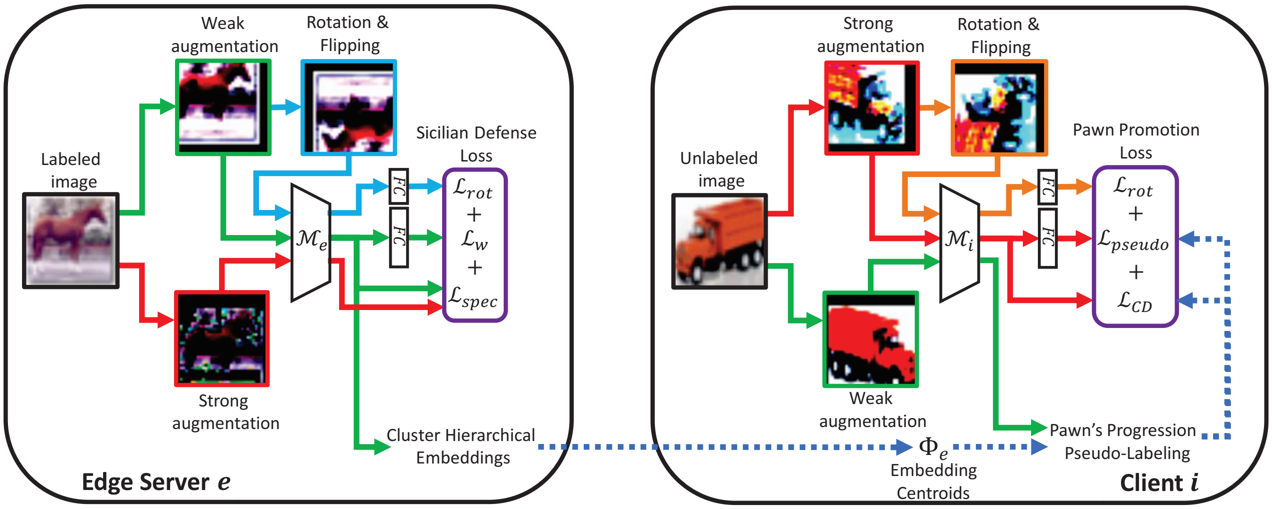

Fig. 2: Our proposed CHESSFL framework initiates training on the edge server with labeled data, utilizing our novel Sicilian Defense Loss. The feature embeddings derived from weakly-augmented labeled images are clustered, and the embedding cluster centroids are then transmitted to the client. Following the implementation of the proposed Pawn's Progression Pseudo-Labeling, the client starts training on unlabeled images using our novel Pawn Promotion Loss.

far too costly and not readily available. Second, since real-world clients usually have unlabeled data, it is unfeasible to consider that labeled data could be simply shared between individuals, yet alone organizations, considering potential privacy violations and proprietary datasets. Therefore, widely available and high-quality labeled data is desirable, but not achievable in the real world. Indeed, since General Data Protection Regulation (GDPR) [8] and California Consumer Privacy Act (CCPA) [9] imposed more constraints on data sharing, decentralized learning at scale has become a critical problem [7]. Therefore, there is a natural synergy between FL and semi-supervised learning (SSL). However, how to leverage unlabeled data in FL is still under-explored. This is because unlabeled data is widely available in FL and too few clients are motivated enough to label their own local data.

As opposed to traditional SSL, in semi-supervised FL (SSFL) the labeled and unlabeled data are split between the cloud and the clients, respectively. This isolation of labeled data in the cloud may compromise the overall performance [7]. Besides, data heterogeneity naturally occurs in FL scenarios under the non-IID umbrella, therefore balancing the performance and the communication efficiency of SSFL is of paramount importance.

In cross-device FL there are two scenarios discussed in [10] based on where labeled data resides: labels-at-client or labels-at-server. Consequently, in cross-device SSFL, the most common scenario is *labels-at-server*, with all clients having large amounts of unlabeled data and the cloud having access to limited labeled data. This is not only more challenging, but also more realistic since the vast majority of clients do not want to spend time and resources labeling their own data or may not even have the proper expertise to do so [7].

Considering the challenges posed by realistic HFL at scale and the issue of implementing SSL in a real-world decentralized setting, we address the following research questions:

- Can we leverage the client-edge-cloud hierarchy to perform better semi-supervised learning in hierarchical federated learning with minimal communication overhead?
- How much better does an approach that leverages the client-edge-cloud hierarchy perform in terms of convergence speed and accuracy when compared to other methods in semi-supervised hierarchical federated learning?

To address the first research question, we propose Clustering Hierarchical Embeddings for Semi-Supervised Federated Learning (CHESSFL) as a solution that benefits from the hierarchical structure of the system to speed up the learning convergence and increase the overall accuracy. To achieve this, we combine unsupervised learning techniques such as rotation prediction [11]–[13] with semi-supervised learning techniques like pseudo-labeling and propose two new loss functions.

To answer the second research question, we propose a unified semi-supervised HFL (SSHFL) framework, depicted in Fig. 1, to run experiments that evaluate how clients learn in a semi-supervised and hierarchical system. Our unified framework first partitions the dataset of choice into labeled and unlabeled datasets. The labeled data is split in an IID fashion among all edge servers and the remaining unlabeled partition of the dataset is split in either IID or non-IID fashion among all clients. This way, each edge server has access only to its own labeled data, and each client has access to its own unlabeled dataset. We then run the classical HFL learning scheme shown in Fig. 1 by steps 1 through 5. Finally, we evaluate the convergence speed to a given accuracy threshold and test accuracy and test loss of the global model.

Despite the limited existing research in SSFL, the more realistic and scalable SSHFL is not yet explored. Similar to Queen's Gambit chess opening, we move to the center of this new research area by providing (i) CHESSFL, the first solution designed for SSHFL, and (ii) a unified evaluation framework designed specifically for SSHFL. We outline our proposed CHESSFL framework in Fig. 2.

Our contributions are as follows:

- **Clustering Hierarchical Embeddings for Semi-Supervised Federated Learning (CHESSFL)**: In order to leverage the hierarchy of the system, we propose a novel clustering mechanism for hierarchical embeddings to improve convergence speed and performance in terms of accuracy for SSHFL. To the best of our knowledge, this solution is the first to account for both the client-edge-cloud hierarchy in FL and SSL.
- **Unified Semi-Supervised Hierarchical Federated Learning Framework**: To the best of our knowledge, we are the first to propose a unified framework for SSHFL and evaluate the performance of straightforward implementations of current state-of-the-art SSFL algorithms in the hierarchical setting.
- **Empirical Validation**: We show that CHESSFL converges up to $5.11\times$ faster and achieves higher accuracy than state-of-the-art SSFL solutions on SVHN, CIFAR10 and CIFAR100 datasets, with negligible communication overhead and enhanced robustness to non-IID data.

The remainder of the paper is organized as follows: Section II reviews the relevant prior work. Section III introduces our proposed approach. The experimental results are presented in Section IV. Finally, Section V summarizes our key findings and outlines directions for future work.

II. Related Work: Historical Matches

Hierarchical Federated Learning: The classical client-edge-cloud HFL scenario is depicted for the first time by Liu et al. [3], where they use 50 edge devices and 5 edge servers. HierFAVG [3] shows that HFL using FedAVG [1] is an effective way of reducing training time and energy consumption of edge devices across the board compared to cloud-based FL. In [6] the authors extend the original HFL work by introducing differential privacy in the hierarchy. Abad et al. [14] extend the original HFL work for communication efficient HFL in the heterogeneous cellular networks. Similarly, Yuan et al. [15] propose a new communication protocol that benefits from local-area network instead of only using a wide-area network to further improve communication efficiency and speed up training for HFL. MACFL [16] is an extension of HierFAVG towards simulating user mobility.

All previously discussed HFL works build upon the foundation of HierFAVG by improving communication and computation efficiency, and by enabling HFL to run in more realistic scenarios. Our paper complements all these works by opening another new direction for HFL, *i.e.,* SSHFL.

Semi-Supervised Learning: The goal of SSL is to learn from unlabeled samples with limited labeled samples, assuming the same distribution between labeled and unlabeled data. A common approach is to use pseudo-labels [17] generated with a model pre-trained on the labeled data to learn the unlabeled data as a form of entropy regularization. FixMatch [18] marked the inception of a trend in SSL. This approach predicts the class of a weakly-augmented image and if the initial prediction is over a certain threshold, uses the predicted class as a pseudo-label in the cross-entropy loss for the prediction of the strongly-augmented version of the same image. As setting a rigid threshold for pseudo-labeling lacks flexibility, FlexMatch [19] enables a flexible threshold based on curriculum pseudo-labeling. FullMatch [20] combines FixMatch with negative learning, leveraging all the other class probabilities, besides the one used for pseudo-labeling. Similarly, MaxMatch [21] minimizes the worst-case consistency between the original sample and its augmented versions, resulting in a more robust model for SSL.

Apart from pseudo-labeling, another SSL approach is Mean Teacher [22] which leverages two models, *i.e.,* teacher and student. The student model is training to be consistent with the teacher model, while the teacher model is updated with an exponentially moving average technique, with very small updates from the student model every iteration.

Another line of research in SSL focuses on how to leverage unsupervised (*i.e.,* self-supervised) learning techniques to boost learning. One of the first approaches in this area was S^4L [23] where the authors propose to unify self-supervised and semi-supervised approaches showing how to train models to achieve new state-of-the-art results on semi-supervised ILSVRC-2012 dataset [24]. USADTM [25] uses a triplet mutual information (MI) loss for unsupervised learning and a deformable template matching to align clustering labels learned from MI, continuously optimizing the feature distribution of labeled data. RotNet [12] performs random rotations for $\{0°, 90°, 180°, 270°\}$ for unsupervised learning and is extended by SESEMI [13] for SSL with vertical and horizontal flips of the image besides the aforementioned rotations.

The pseudo-labeling based works use the pseudo-labels combined with consistency regularization (*i.e.,* consistent predictions of the same image augmented in different ways) and achieve good performance for SSL. Conversely, the SSL works that leverage unsupervised learning techniques not only yield good performance, but also have better and more robust representations. The critical issue is that all previously discussed approaches consider both labeled and unlabeled losses in a centralized server, which is not achievable in HFL given the data privacy concerns.

Semi-Supervised Federated Learning: The adaptation of SSL approaches to FL is not trivial. For example, FedMatch [10] uses two sets of parameters for learning labeled and unlabeled data, respectively. For pseudo-labeling, FedMatch uses an agreement-based pseudo-label by sending to a client the top-k most similar models from other clients selected by the central server and maximizes their agreement for the pseudo-label, therefore introducing a communication overhead. SemiFL [26] fine-tunes the global model with the labeled data to alleviate the forgetting effect (*i.e.,* decreasing accuracy over time), observed in FedMatch, caused by training solely on unsupervised data from the clients. The local models of the clients are trained on pseudo-labels generated by the global model on the client-side unlabeled data, obtaining more competitive results compared to FedMatch. SemiFL uses static batch normalization (sBN) [27] to initialize the running mean and running variation parameters from the batch normalization operations in their models. Furthermore, SemiFL uses both labeled and unlabeled datasets to compute the sBN statistics, which is an unrealistic assumption. Similar to FixMatch, SemiFL builds a training dataset called fixed dataset using weakly-augmented images if their prediction probabilities are above a given threshold. With this fixed dataset, SemiFL builds another dataset, called mixed dataset, by randomly sampling two images from the fixed dataset and mixing them using MixMatch [28]. This introduces a computational overhead and increases memory usage for every client, since both fixed and mixed datasets are built and used for training on-device.

Orchestra [29] is an *unsupervised* approach that combines the rotation prediction with the Mean Teacher approach to train better representations on the unlabeled clients. By using the Mean Teacher approach, Orchestra introduces a computational overhead on the resource-limited clients while also consuming more energy to keep both teacher and student models running on-device. The clustering introduced in [29] is not actual clustering, but rather extra classification heads, *i.e.,* linear layers, for local and global clusters, respectively, while considering as "cluster centroids" the weights of the respective linear layers. Even so, when using ResNet18, a model almost 10× larger compared to Wide ResNet 28x2 used by SemiFL, Orchestra still falls short in its semi-supervised evaluation compared to SemiFL. Furthermore, since SemiFL is a SSFL approach, we compare our work to SemiFL instead of Orchestra.

Recent FL survey papers such as [7] stress that the problem of semi-supervised FL is under-explored and needs better solutions that strike a right balance between the performance and efficiency under heterogeneity constraints. Indeed, seeing the lack of semi-supervised type of research in HFL, we aim to bridge this gap and provide a baseline for SSHFL to encourage research in this more practical and realistic side of FL.

III. Methodology: Defense and Counterplay

Chessboard Strategies: We first propose the Sicilian Defense Loss to train the edge servers on labeled data and enhance their capability to have diverse and robust feature embeddings. The resulted feature embeddings are then clustered and shared throughout the hierarchy with the clients, which use the embedding centroids in Pawn's Progression Pseudo-Labeling. Lastly, we introduce the Pawn Promotion Loss, which facilitates the convergence of client embeddings toward the embedding centroids and ensures the learning of diverse feature embeddings on the client side as well.

Choosing the Pieces: In our SSHFL setting, we consider two datasets: a labeled dataset $\mathbb{L}$ and an unlabeled dataset $\mathbb{U}$, with $|\mathbb{U}| \gg |\mathbb{L}|$ and $|\cdot|$ denoting the size of a set. We also assume the marginal distributions of $\mathbb{L}$, $\mathbb{U}$ are the same. Previous SSFL setups included the labeled dataset in the cloud and the unlabeled dataset split either IID or non-IID between all devices. We adapt the labeled dataset $\mathbb{L}$ for HFL by splitting it in an IID manner among all edge servers:

$$\bigcup_{e \in \mathbb{E}} \mathbb{L}_e = \mathbb{L}, \text{ with } |\mathbb{L}_i| = |\mathbb{L}_j|, \forall i, j \in \mathbb{E}, i \neq j \tag{1}$$

with $\mathbb{E}$ being the total number of edge servers and $\mathbb{L}_e$ the labeled dataset available at edge server e. We argue that in a realistic HFL environment it makes sense to put labeled data on the edge servers instead of the cloud, since edge servers are physically closer to the edge devices. This proximity facilitates faster localized training and offers advantages such as operational efficiency, adaptability, and cost-effectiveness. We assume each edge server e can only access its own labeled dataset $\mathbb{L}_e$.

The unlabeled dataset $\mathbb{U}$ is split between all clients IID and non-IID, using $\mathbb{U}_i \sim Dir(\alpha)$ with $\mathbb{U}_i$ as the unlabeled local dataset available at device i, where $i \in \mathbb{D}$ and $\mathbb{D}$ is the set that contains all devices, with $Dir(\alpha)$ as the Dirichlet distribution [30] to sample IID and non-IID datasets. We note that in this SSHFL setting, each client i only has access to its own unlabeled dataset $\mathbb{U}_i$.

Opening Principles: Given a training dataset $\mathbb{U}_i = \{x_u\}_{u=1}^{|\mathbb{U}_i|}$ for device i with $|\mathbb{U}_i|$ as the total number of training samples for device i and x_u the u^{th} training sample, in FL we want to solve the following optimization problem:

$$\min_{\theta_{\mathcal{G}}} f(\theta_{\mathcal{G}}) = \frac{1}{|\mathbb{D}|} \sum_{i=1}^{|\mathbb{D}|} \mathcal{L}_u(\theta, \mathbb{U}_i) \tag{2}$$

where $\mathcal{L}_u$ is the loss function for unsupervised data evaluated on the local dataset $\mathbb{U}_i$ using the local model weights θ_i for client i, with $\theta_{\mathcal{G}}$ being the global weights and f being the global loss function. In FL we solve the optimization problem using the cloud and the clients, while in HFL, we also use edge servers. For practical purposes, the optimization problem we solve for SSHFL follows the same principles defined in SSL:

$$\min_{\theta_{\mathcal{G}}} f(\theta_{\mathcal{G}}) = \frac{1}{|\mathbb{E}|} \sum_{e=1}^{|\mathbb{E}|} \mathcal{L}_s(\theta_e, \mathbb{L}_e) + \frac{1}{|\mathbb{D}|} \sum_{i=1}^{|\mathbb{D}|} \mathcal{L}_u(\theta_i, \mathbb{U}_i) \tag{3}$$

with $\mathcal{L}_s$ as the loss for supervised data and θ_e as the edge model weights for edge server e. Edge servers have only labeled data and clients have only unlabeled data. Each edge server has its own labeled training dataset $\mathbb{L}_e = \{x_l, y_l\}_{l=1}^{|\mathbb{L}_e|}$ with $|\mathbb{L}_e|$ as the total number of labeled training samples for edge server e, and x_l as the l^{th} training sample with its corresponding label y_l.

In order to solve Eq. 3, we propose CHESSFL, detailed in Alg. 1, which has two innovative components: the *Sicilian Defense Loss (SDLoss)* for training on edge servers with labeled data and the *Pawn Promotion Loss (PawnLoss)* for training on the clients with pseudo-labeled data. Finally, we unify the SSHFL framework using a more flexible aggregation technique both at the edge server and in the cloud.

Algorithm 1 Clustering Hierarchical Embeddings for Semi-Supervised Federated Learning (CHESSFL)

```
Initialize global weights $\theta_{\mathcal{G}}$ with random weights and download them on all edge servers $\theta_e \leftarrow \theta_{\mathcal{G}}, \forall e \in \mathbb{E}$
$\theta_e$, $\Phi_e \leftarrow$ Train&ClusterEdgeServer($e$), $\forall e \in \mathbb{E}$    ▷ Get trained weights $\theta_e$ and embedding centroids $\Phi_e$
for communication round $k$ = 1, 2,..., $K$ do
    for each device $i \in \mathbb{D}$ in parallel do
        Download the latest edge model weights $\theta_i \leftarrow \theta_e$ and embedding centroids $\Phi_e$
        Initialize the pseudo-labeled dataset $\mathbb{U}_i^* = \emptyset$
        for $x_u \in \mathbb{U}_i$ do    ▷ For all unlabeled images $x_u \in \mathbb{U}_i$
            $z_w^u = \mathcal{M}_i(\mathcal{W}(x_u))$    ▷ Obtain the feature embeddings $z_w^u$ from client model $\mathcal{M}_i$ using the
                                                         weakly-augmented unlabeled sample $\mathcal{W}(x_u)$
            $z_w^{u,norm} = z_w^u / \|z_w^u\|_2$    ▷ Normalize the feature embeddings
            Compute class labels $\hat{c}_{cos}$ and $\hat{c}_{euc}$ based on Eq. 19, 20 with $\phi_c \in \Phi_e$ and $z_w^{u,norm}$
            if $\hat{c}_{cos} = \hat{c}_{euc}$ and $cos(z_w^{u,norm}, \phi_c) \geq \tau$ then
                $y_u^*, \phi^* \leftarrow \hat{c}_{cos}, \phi_c$
                $\mathbb{U}_i^* = \mathbb{U}_i^* \cup \{x_u, y_u^*\}$
            end if
        end for
        $\theta_i \leftarrow$Train($\mathcal{L}_{Pawn}$, $\mathbb{U}_i^*$, $k_1$, $\theta_i$)    ▷ Train the parameters $\theta_i$ of the client model $\mathcal{M}_i$ for $k_1$ local epochs using the
                                                         Pawn Promotion Loss $\mathcal{L}_{Pawn}$ from Eq. 21 on the pseudo-labeled dataset $\mathbb{U}_i^*$
    end for
    $\theta_e = \sum_{i \in \mathbb{D}_e} p_i^e \theta_i$, $\forall e \in \mathbb{E}$    ▷ Aggregate local model weights $\theta_i$ for all clients $i$ connected to edge server $e$ using Eq. 25
    $\theta_e$, $\Phi_e \leftarrow$ Train&ClusterEdgeServer($e$), $\forall e \in \mathbb{E}$
    if $k$ mod $k_2 = 0$ then
        $\theta_{\mathcal{G}} = \sum_{e \in \mathbb{E}} p_e \theta_e$    ▷ Aggregate edge model weights $\theta_e$ using Eq. 26
        $\theta_e \leftarrow \theta_{\mathcal{G}}$, $\forall e \in \mathbb{E}$    ▷ Download updated global model on all edge servers
        $\theta_e$, $\Phi_e \leftarrow$ Train&ClusterEdgeServer($e$), $\forall e \in \mathbb{E}$
    end if
end for
function Train&ClusterEdgeServer($e$)
    $\theta_e \leftarrow$ Train($\mathcal{L}_{SD}$, $\mathbb{L}_e$, $k_e$, $\theta_e$)    ▷ Train the parameters $\theta_e$ of the edge server model $\mathcal{M}_e$ for $k_e$ local epochs
                                                         using the Sicilian Defense Loss $\mathcal{L}_{SD}$ from Eq. 12 on the labeled dataset $\mathbb{L}_e$
    Obtain embedding centroids $\Phi_e$ using the labeled dataset $\mathbb{L}_e$ based on Eq. 15
    return $\theta_e$, $\Phi_e$
end function
```

A. Training and Clustering Embeddings on Edge Servers

Sicilian Defense Loss: In chess, the Sicilian Defense represents a robust foundation, providing significant flexibility, while having a double-edged nature that offers an adaptive evolution throughout the game. We construct the SDLoss function with the same characteristics in mind. We first consider the double-edged nature represented by weakly- and strongly-augmented images. Next, we enable an adaptive evolution of feature embeddings using a variation of rotation prediction [11], [12], which includes horizontal and vertical flipping as well [13]. This enables the edge model to learn better and more robust feature embeddings with enhanced generalization capabilities. Finally, we use the cross-entropy loss between the predicted probabilities from the weakly-augmented image and the ground-truth label.

Let $\mathcal{W}(\cdot)$ represent a weak augmentation composed of random horizontal flipping and random cropping with padding and let $\mathcal{S}(\cdot)$ represent a strong augmentation, *i.e.,* RandAugment [31], on top of random horizontal flipping and random cropping with padding. First, considering a labeled image sample $\{x_l, y_l\}$ and an edge model $\mathcal{M}_e$ for edge server e, we obtain the following:

$$\hat{y}_w, z_w = \mathcal{M}_e(\mathcal{W}(x_l)) \tag{4}$$

$$z_s = \mathcal{M}_e(\mathcal{S}(x_l)) \tag{5}$$

where $\hat{y}_w$ is the weakly-augmented image prediction, and z_w and z_s are the weakly- and strongly-augmented image's resulting Z-dimensional feature embeddings.

Let $\mathcal{R}(\cdot, r)$ be our rotation function that rotates an image by r degrees if $r \in \{0°, 90°, 180°, 270°\}$ or flips an image horizontally or vertically if $r \in \{h, v\}$. Therefore, we define an extra prediction head for our model $\mathcal{M}_e$ with 6 neurons to predict the rotation class performed on the weakly-augmented image as shown in Eq. 6. This is not only a cost-effective method to improve feature robustness, but also a regularization technique. Given that r is randomly selected for every image, it ensures an adaptive evolution of the feature embeddings learned on the limited labeled dataset.

$$\hat{r}_w = \mathcal{M}_e(\mathcal{R}(\mathcal{W}(x_l), r)) \tag{6}$$

Consequently, we build the SDLoss to have flexible yet robust embeddings fit for pseudo-labeling unlabeled data. For this purpose, we first consider the cross-entropy loss for predicting weakly-augmented image class (Eq. 7) and the cross-entropy loss for predicting the rotation of the weakly-augmented and rotated image (Eq. 8).

$$\mathcal{L}_w = \mathbb{H}(\hat{y}_w, y_l) \tag{7}$$

$$\mathcal{L}_{rot}^w = \mathbb{H}(\hat{r}_w, r) \tag{8}$$

In order to form well defined clusters of embeddings, we use the spectral constrastive loss $\mathcal{L}_{spec}$ [32] to create in the embedding space tightly coupled and diverse embedding clusters for each class. For an embedding pair (z_w, z_s) we first normalize each embedding if $\|z\|_2 \geq 1$, where $\|\cdot\|_2$ is the L2 norm. We split the spectral constrastive loss into two components in Eq. 9. The first component, $\mathcal{L}_{same}$, pulls embeddings from augmentations of the same image closer together and the second component, $\mathcal{L}_{diff}$, pushes further apart feature embeddings from images with different labels.

$$\mathcal{L}_{spec} = \mathcal{L}_{same} + \mathcal{L}_{diff} \tag{9}$$

Formally, $\mathcal{L}_{same}$ computes the loss between embeddings from the same image:

$$\mathcal{L}_{same} = -2 \times \frac{1}{B} \sum_{b=1}^{B} \langle z_w^b, z_s^b \rangle \times Z \tag{10}$$

with B as the batch size, $\langle z_w^b, z_s^b \rangle$ as the dot product between the b^{th} embeddings of z_w and z_s and Z as the dimensionality of the embeddings. Consequently, $\mathcal{L}_{diff}$ computes the loss between embeddings from images with different labels from the same batch:

$$\mathcal{L}_{diff} = \frac{1}{B(B-1)} \sum_{b_1=1}^{B} \sum_{\substack{b_2=1 \\ b_2 \neq b_1}}^{B} (\langle z_w^{b_1}, z_s^{b_2} \rangle)^2 \tag{11}$$

where we sum up the squared dot products of embeddings from off-diagonal elements and average over all possible pairs of different embeddings in the batch.

To summarize, we build the Sicilian Defense Loss, a cohesive and robust loss for learning on the labeled data from edge servers (Eq. 12). This loss is meant to take into account the embeddings from the edge model and cluster them accordingly in order to improve the representational capabilities of the feature embeddings.

$$\mathcal{L}_{SD} = \mathcal{L}_w + \mathcal{L}_{rot}^w + \mathcal{L}_{spec} \tag{12}$$

Clustering Embeddings: After training on the edge, we compute the centroids of the embeddings $\mathbb{Z}_c$ for every class $c \in \mathbb{C}$ as follows:

$$\mathbb{Z}_c = \{z_w = \mathcal{M}_e(\mathcal{W}(x_l)), \forall \{x_l, y_l\} \in \mathbb{L}_e, y_l = c\} \tag{13}$$

where $\mathbb{Z}_c$ is a set with all feature embeddings for every weakly-augmented labeled sample in $\mathbb{L}_e$ that has the label $y_l = c$ and $\mathbb{C}$ is the total number of classes. Then, we normalize each feature embedding $\mathbb{Z}_c$ using the L2 norm:

$$\mathbb{Z}_c^{norm} = \left\{ \frac{z_w}{\|z_w\|_2}, \forall z_w \in \mathbb{Z}_c \right\} \tag{14}$$

Finally, we compute embedding centroid ϕ_c for every class $c \in \mathbb{C}$ by averaging all normalized feature embeddings $\mathbb{Z}_c^{norm}$.

$$\phi_c = \frac{1}{|\mathbb{Z}_c^{norm}|} \sum_{\mathbb{Z}_c^{norm}} z_w^{norm} \tag{15}$$

We define $\Phi_e = \{\phi_c, \forall c \in \mathbb{C}\}$ as the set with all embedding centroids from edge server e for all classes $c \in \mathbb{C}$. After training and clustering (Lines 27-28 in Alg. 1), we send to each client of edge server e the edge model parameters θ_e and the embedding centroids set Φ_e (Line 5 in Alg. 1). Clustering embeddings enables the class-wise average of features learned on labeled data from each edge server to be used as embedding centroids for pseudo-labeling on the clients.

B. Pseudo-Labeling and Training on Edge Devices

Pawn's Progression Pseudo-Labeling: Since pawns in chess can only move forward and attack in diagonal, we use this as inspiration for pseudo-labeling. Considering an unlabeled local image x_u and the local model $\mathcal{M}_i$ for client i, we obtain the following predictions and feature embeddings:

$$z_w^u = \mathcal{M}_i(\mathcal{W}(x_u)) \tag{16}$$

$$\hat{y}_s^u, z_s^u = \mathcal{M}_i(\mathcal{S}(x_u)) \tag{17}$$

$$\hat{r}_s^u = \mathcal{M}_i(\mathcal{R}(\mathcal{S}(x_u), r)) \tag{18}$$

where $\hat{y}_s^u$, z_s^u are the predicted class and the corresponding feature embedding for the strongly-augmented unlabeled sample x_u, $\hat{r}_s^u$ is the prediction of the rotation for the strongly-augmented training sample x_u and z_w^u is the feature embedding obtained from the weakly-augmented unlabeled sample x_u.

Similar to a pawn's progression being forward with diagonal attacking options, we pseudo-label based on a "direct" Euclidean distance and "diagonal" cosine similarity between the local embeddings and the embedding centroids from the edge server. This way we provide a comprehensive view of embedding relationships, ensuring a more accurate and nuanced pseudo-labeling [25]. First, we normalize the local embeddings of the weakly-augmented images $z_w^{u,norm} = z_w^u / \|z_w^u\|_2$. Then, we compute the class $\hat{c}_{cos}$ that maximizes the cosine similarity and the class $\hat{c}_{euc}$ that minimizes the Euclidean distance by finding the class $c \in \mathbb{C}$ of the closest embedding centroid $\phi_c \in \Phi_e$ as follows:

$$\hat{c}_{cos} = \underset{c}{\operatorname{argmax}} \ \cos(z_w^{u,norm}, \phi_c) \tag{19}$$

$$\hat{c}_{euc} = \underset{c}{\operatorname{argmin}} \ \mathrm{d}(z_w^{u,norm}, \phi_c) \tag{20}$$

where $\cos(x, y) = \frac{\langle a,b \rangle}{\|a\|\|b\|}$ and $\mathrm{d}(x, y) = \sqrt{(x-y)^2}$ are the cosine similarity and Euclidean distance functions, respectively. This way, if $\hat{c}_{cos} = \hat{c}_{euc}$ and if the maximum cosine similarity is above a predefined confidence threshold τ, we select the embedding centroid ϕ^* and its corresponding pseudo-label y_u^* as the selected class $\hat{c}_{cos}$ (Line 12 in Alg. 1). In case $\hat{c}_{cos} \neq \hat{c}_{euc}$, we disconsider the unlabeled training sample x_u. Consequently, we have a pseudo-labeled training set $\mathbb{U}_i^* \subseteq \mathbb{U}_i$ using only the unlabeled images $x_u \in \mathbb{U}_i$ that were assigned a pseudo-label y_u^* (Line 13 in Alg. 1).

Pawn Promotion Loss: Similar to how pawns eventually get promoted by reaching the opposite end of the chessboard, we want to enforce local learning of the high confidence embedding centroids for unlabeled training samples to enhance the local pseudo-labeling. We adapt the rationale behind FixMatch which uses weakly-augmented predictions as ground-truth for learning strongly-augmented predictions. As a result, we use a continuous optimization process where clients use high confidence embedding centroids obtained on weakly-augmented labeled images from the edge server to create pseudo-labels from the weakly-augmented unlabeled local images (Lines 5-15 in Alg. 1). These pseudo-labels are then used as ground-truth for learning the strongly-augmented unlabeled images (Line 16 in Alg. 1). Since all clients are encouraged to push their own local embeddings closer to the chosen embedding centroids from the edge server, this continues to increase the confidence for the respective class label. For this, we use the PawnLoss defined as follows:

$$\mathcal{L}_{Pawn} = \mathcal{L}_{pseudo} + \mathcal{L}_{rot}^{s} + \mathcal{L}_{CD} \tag{21}$$

$$\mathcal{L}_{pseudo} = \mathbb{H}(\hat{y}_s^u, y_u^*) \tag{22}$$

$$\mathcal{L}_{rot}^{s} = \mathbb{H}(\hat{r}_s^u, r) \tag{23}$$

where $\mathcal{L}_{CD}$ is the Centroid Distillation Loss using the Kullback-Leibler divergence on the softmax probabilities of the local embeddings z_s^u and the selected embedding centroid ϕ^* defined as follows:

$$\mathcal{L}_{CD} = D_{KL}(softmax(z_s^u/T) \| softmax(\phi^*/T)) \tag{24}$$

with T as the temperature to smooth the probabilities of the feature embeddings [33]. This enables us to continuously refine the local embeddings with the guidance of embedding centroids received from the edge server.

C. Hierarchical Embeddings and their Aggregation

Instead of averaging the weights like HierFAVG [3], we weight the contribution of each model in the aggregation using cosine similarity-based weights. We define the aggregation of local model parameters θ_i at edge server e as follows:

$$\theta_e = \sum_{i \in \mathbb{D}_e} p_i^e \theta_i, \text{ where } p_i^e = \frac{e^{-\sigma \cos(\theta_e,\, \theta_i)}}{\sum\limits_{j \in \mathbb{D}_e} e^{-\sigma \cos(\theta_e,\, \theta_j)}} \tag{25}$$

where $\mathbb{D}_e \subset \mathbb{D}$ is the subset of devices connected to edge server e, and σ is a hyperparameter. Likewise, for global aggregation we use:

$$\theta_{\mathcal{G}} = \sum_{e \in \mathbb{E}} p_e \theta_e, \text{ where } p_e = \frac{e^{-\sigma \cos(\theta_{\mathcal{G}},\, \theta_e)}}{\sum\limits_{j \in \mathbb{E}} e^{-\sigma \cos(\theta_{\mathcal{G}},\, \theta_j)}} \tag{26}$$

As we learn embeddings using the SDLoss for better clusterability and enable clients to learn the embedding centroids when confident enough, all these embeddings must be shared throughout the hierarchy for the learning process to be successful. Using the cosine similarity based aggregation both at the edge (Line 18 in Alg. 1) and in the cloud (Line 21 in Alg. 1), we ensure that the embeddings, when shared throughout the hierarchy, are still relevant for the downstream clustering and pseudo-labeling.

On top of this, after the edge and global aggregations, we fine-tune the aggregated edge models on the available labeled data at each edge server. Finally, we cluster the hierarchical embeddings to send new embedding centroids to the edge devices (Lines 19 and 23 in Alg. 1).

IV. Experimental Results: Moves and Consequences

A. Experimental Setup - Board Battle Plan

The Wooden Pieces: We evaluate SSHFL by simulating all our experiments on three GPU servers, two identical ones, each with 4× NVIDIA RTX A6000 GPUs, a 64-core AMD Threadripper PRO 3995WX CPU and 512GB RAM, and a larger GPU server with 8× NVIDIA RTX A6000 Ada generation GPUs, 2× 32-core AMD EPYC 7513 CPUs and 1TB RAM. We simulate the client-edge-cloud hierarchy for all experiments to provide performance evaluations in terms of global test accuracy and global test loss.

The Strategy: For all experiments, we use the Wide ResNet 28x2 model [34], since most related SSL and SSFL approaches use it [18]–[20], [25], [26]. In order to follow standard semi-supervised procedure, we choose the top-3 datasets used in this area, *i.e.,* CIFAR10, CIFAR100 [35] and SVHN [36]. We consider 4,000 labeled images for CIFAR10 and SVHN and 10,000 labeled images for CIFAR100. We establish as standard for SSHFL that the edge servers split among themselves the labeled dataset, in an IID manner. Splitting in an IID manner the labeled data on the edge servers boosts the learning of the unlabeled clients. The remaining unlabeled images are then split among all clients by using the Dirichlet distribution [30]. For IID distribution, images are sampled with $\alpha = 100$ and $\alpha = 0.1$ is used for non-IID distribution. In order to balance the unlabeled data for all clients, we assume each client has 500 unlabeled images, since we consider clients as edge devices with limited memory and computational capabilities. This approach ensures consistent evaluations across all datasets. Given that different datasets have varying total numbers of images, *e.g.,* CIFAR10 has 50,000 images and SVHN contains 73,257, it is crucial to maintain the same number of images per client. Therefore, we ensure a realistic evaluation of SSHFL across all datasets. This is an extension of the labels-at-server scenario to the SSHFL setting which we simply call labels-at-edge.

Similar to the original HFL system proposed in [3], we consider 5 edge servers and 50 edge devices (i.e., clients), with each server having access to an equal number of clients. Following HierFAVG [3], we assume all clients are always

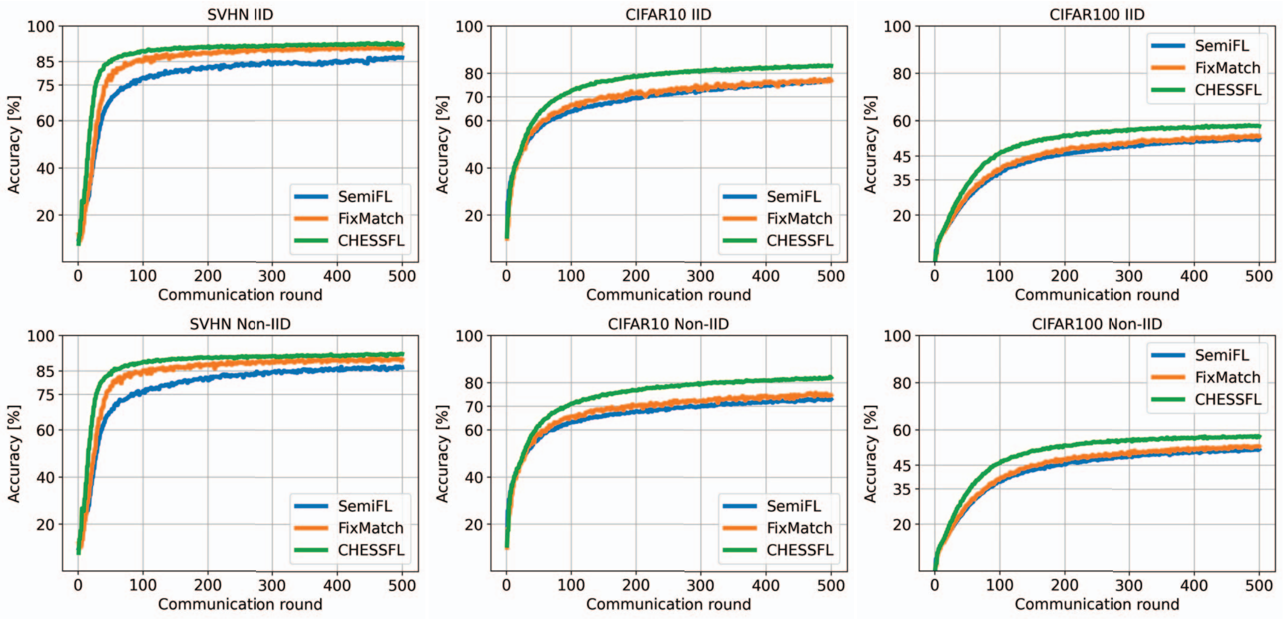

Fig. 3: Global test accuracy results using IID and non-IID settings for $k_1 = 1$, $k_e = 1$ and $k_2 = 2$. Overall, we observe that CHESSFL converges faster and reaches higher accuracies for all datasets, in both IID and non-IID settings.

TABLE I: Global test accuracies [%] for $k_1 = 1$, $k_e = 1$ and variations of k_2 in IID and non-IID settings. The effectiveness of clustering hierarchical embeddings enables CHESSFL to obtain higher accuracy values in all scenarios.

Dataset		IID ($\alpha = 100$)			Non-IID ($\alpha = 0.1$)		
	k_2	2	5	10	2	5	10
SVHN	SemiFL	86.81 ± 0.37	87.52 ± 1.18	87.83 ± 1.33	86.84 ± 0.67	87.56 ± 0.01	87.37 ± 0.06
	FixMatch	91.22 ± 0.24	91.60 ± 0.77	91.74 ± 0.23	90.33 ± 0.31	90.84 ± 0.02	90.92 ± 0.01
	CHESSFL	**92.84 ± 0.12**	**92.54 ± 0.04**	**92.12 ± 0.41**	**92.24 ± 0.22**	**92.11 ± 0.25**	**91.81 ± 0.20**
CIFAR10	SemiFL	77.13 ± 0.30	76.44 ± 0.08	76.76 ± 0.74	73.43 ± 0.52	74.11 ± 0.46	73.64 ± 0.88
	FixMatch	77.57 ± 0.38	77.82 ± 0.07	77.85 ± 0.15	75.68 ± 0.21	75.55 ± 0.32	75.80 ± 0.33
	CHESSFL	**83.29 ± 0.36**	**82.97 ± 0.02**	**81.95 ± 0.09**	**82.46 ± 0.23**	**81.37 ± 0.35**	**80.44 ± 0.08**
CIFAR100	SemiFL	52.27 ± 0.43	50.70 ± 0.30	48.95 ± 0.07	51.74 ± 0.07	51.47 ± 0.27	48.53 ± 0.30
	FixMatch	53.58 ± 0.22	52.29 ± 0.67	50.78 ± 0.02	53.14 ± 0.54	52.75 ± 0.29	50.65 ± 0.15
	CHESSFL	**57.92 ± 0.24**	**56.40 ± 0.12**	**55.03 ± 0.08**	**57.32 ± 0.24**	**56.23 ± 0.13**	**54.04 ± 0.14**

available and we use during every communication round all 50 clients. We run all experiments for $K = 500$ communication rounds and average all results over 3 different seeds. We use the following hyperparameter values: learning rate of 0.03 for which we use a cosine annealing learning rate scheduler [37], $T = 4$ for Eq. 24, $\sigma = 0.1$ for Eq. 25 and Eq. 26, and the confidence threshold $\tau = 0.7$ (Line 11 in Alg. 1).

We argue that in real-world scenarios, clients prefer to train less due to their limited computational resources and battery life. Besides this, clients can benefit from communicating more with the edge servers and training less locally in HFL. Indeed, as shown in [3], more frequent communication with the edge servers and less local computation, hence lower k_1 values, can speed up training. Therefore, we run all experiments with $k_1 = 1$ and $k_e = 1$, but we also provide ablation studies with different combinations of k_1 and k_e in Section IV-C.

The Opponents: As baselines for SSHFL, we adapt the state-of-the-art SemiFL to HFL and we also implement FixMatch in HFL. We do not use FedMatch due to their communication overhead, the privacy concerns when sharing models from other clients to "help" pseudo-labeling, and low performance compared to SemiFL. We implement FixMatch for every client to run locally, as described in [18]. On top of the implementation from [18], we use fine-tuning with labeled data on the edge server for k_e epochs, similar to SemiFL and CHESSFL. This implementation of FixMatch serves as a straightforward implementation of FixMatch [18] for SSHFL. We implement SemiFL as described in [26] and we only use the labeled data available at each edge server for the computation of the sBN statistics, *i.e.,* running mean and variance. This way, we ensure a more realistic adaptation of SemiFL in the context of SSHFL.

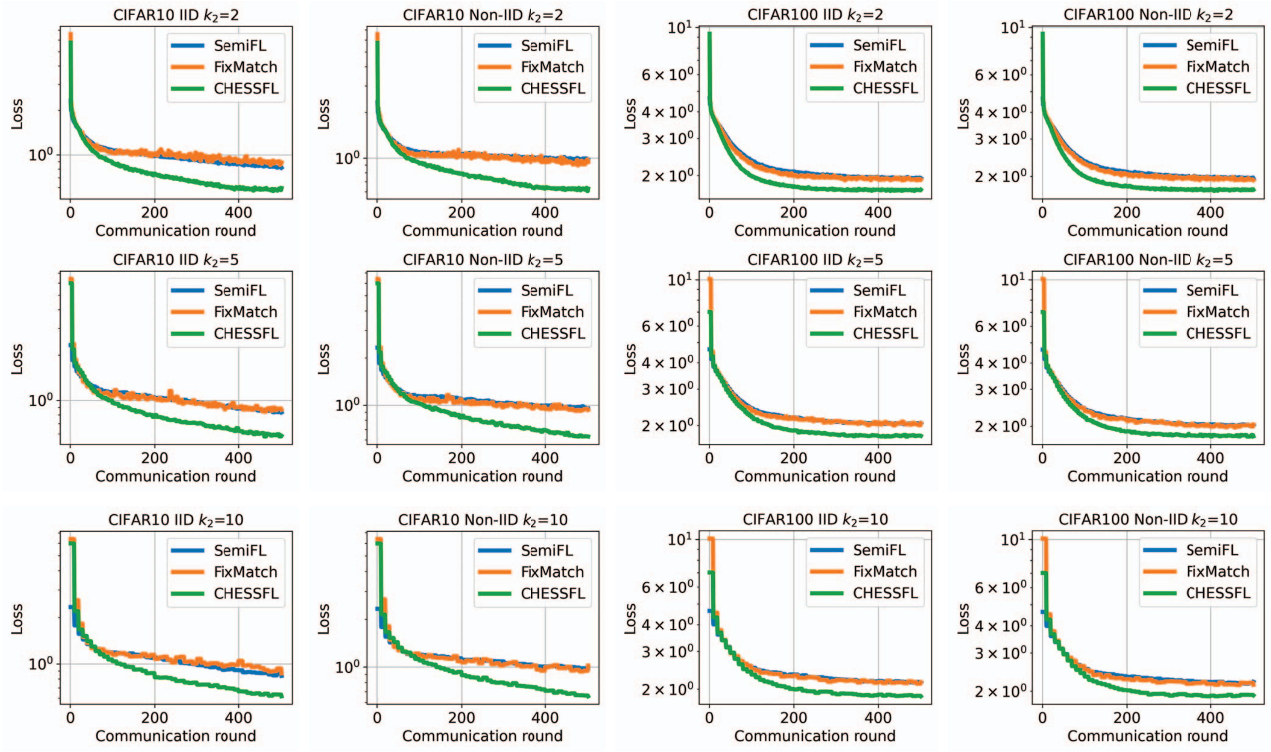

Fig. 4: Global test loss results for $k_1 = 1$, $k_e = 1$ and various k_2 values for CIFAR10 and CIFAR100 datasets in both IID and non-IID settings. CHESSFL consistently demonstrates a more efficient convergence and lower loss across all scenarios.

B. Empirical Results - Tactical Triumphs

Opening Outcome: As we carefully designed CHESSFL specifically for SSHFL using the SDLoss and PawnLoss, we have a unique advantage that makes our method better compared to other, state-of-the-art approaches. In Fig. 3, we observe that CHESSFL learns faster and better compared to the SemiFL and FixMatch baselines. The benefit of using the SDLoss can be seen in how CHESSFL deals with non-IID data, compared to the baselines, *i.e.,* SemiFL and FixMatch. In the context of SSHFL, a straightforward implementation of FixMatch consistently outperforms SemiFL across various scenarios. This performance gap is especially pronounced on the SVHN dataset, where FixMatch demonstrates a notably higher accuracy compared to SemiFL, in both IID and non-IID scenarios. FixMatch only uses a fixed dataset built using all weakly-augmented images that have a prediction over a given confidence threshold. Besides the fixed dataset, SemiFL also uses a mixed dataset created by mixing pairs of images randomly sampled from the fixed dataset. The noise introduced by the mixed dataset in the training process degrades the overall performance of SemiFL across all datasets.

One of the most important hyperparameters in HFL is k_2 since it quantifies the number of edge aggregations to be performed before one global aggregation. We explore multiple variations of this hyperparameter, *i.e.,* $k_2 \in \{2, 5, 10\}$ and show the impact on the global test loss in Fig. 4. Based on data in the first row from Fig. 4 and the accuracy results from the second and third column from Fig. 3 we can indeed confirm that the loss decreases more for CHESSFL for CIFAR10 and CIFAR100 datasets in both IID and non-IID settings. This shows CHESSFL enables a more robust learning overall, throughout multiple communication rounds, compared to the baselines. A more detailed depiction of the impact k_2 has on the learning process can be seen in Table I. For CIFAR10, the average decrease in accuracy from IID to non-IID setting for SemiFL is 3.05% and for FixMatch is 2.07%, while for CHESSFL is only 1.31%. This demonstrates that CHESSFL learns more robust feature embeddings across the client-edge-cloud hierarchy, enhancing overall robustness to non-IID data.

Fast Moves: For all datasets, with varying k_2 values, we show in Table II the number of communication rounds required to achieve a given accuracy threshold. Overall, CHESSFL converges up to 5.11× faster compared to the baselines. This convergence speedup also means CHESSFL enables clients to perform less computation and communication to reach a given accuracy threshold. For example, CHESSFL reaches 70% accuracy on CIFAR10 with IID data in only 83 communication rounds, 2.14× faster, on average, compared to the baselines. Therefore, in IoT systems where energy consumption truly matters and the network connectivity is unreliable, CHESSFL can really make a difference.

TABLE II: Number of communication rounds required to achieve a threshold accuracy (Acc. thresh.) for $k_1 = 1$, $k_e = 1$ and various k_2 values in both IID and non-IID settings. CHESSFL uses up to 5.11× less communication rounds compared to SemiFL and FixMatch. CHESSFL speeds up convergence consistently across both IID and non-IID settings, for all datasets.

Dataset		IID ($\alpha = 100$)			Non-IID ($\alpha = 0.1$)		
	k_2	2	5	10	2	5	10
SVHN Acc. thresh. 85%	SemiFL	393	219	239	309	319	289
	FixMatch	87	89	99	99	119	129
	CHESSFL	**47**	**64**	**79**	**55**	**79**	**89**
	Avg. improvement	**5.11**×	**2.41**×	**2.14**×	**3.71**×	**2.77**×	**2.35**×
CIFAR10 Acc. thresh. 70%	SemiFL	211	239	259	283	264	299
	FixMatch	145	159	179	187	189	189
	CHESSFL	**83**	**99**	**109**	**91**	**114**	**139**
	Avg. improvement	**2.14**×	**2.01**×	**2.01**×	**2.58**×	**1.99**×	**1.76**×
CIFAR100 Acc. thresh. 45%	SemiFL	181	219	269	187	209	299
	FixMatch	151	184	209	157	169	209
	CHESSFL	**93**	**124**	**149**	**93**	**124**	**149**
	Avg. improvement	**1.78**×	**1.63**×	**1.6**×	**1.85**×	**1.52**×	**1.7**×

TABLE III: Global test accuracies for $k_2 = 2$ and variations of k_1 and k_e in IID and non-IID settings. On average, CHESSFL maintains the highest accuracy in both IID and non-IID settings, across all datasets.

Dataset			SVHN			CIFAR10			CIFAR100		
	k_1	k_e	SemiFL	FixMatch	**CHESSFL**	SemiFL	FixMatch	**CHESSFL**	SemiFL	FixMatch	**CHESSFL**
IID ($\alpha = 100$)	1	1	87.91%	91.96%	**93.06%**	77.34%	77.92%	**83.61%**	52.86%	54.09%	**57.95%**
	1	5	90.77%	92.89%	**93.79%**	82.28%	82.14%	**84.28%**	53.79%	55.28%	**55.67%**
	5	1	91.75%	93.81%	**94.44%**	**89.07%**	82.22%	85.93%	52.17%	54.28%	**58.66%**
	5	5	91.25%	93.70%	**94.07%**	**87.62%**	83.16%	85.48%	53.22%	55.42%	**56.32%**
Non-IID ($\alpha = 0.1$)	1	1	87.62%	90.73%	**92.21%**	73.62%	76.01%	**82.81%**	52.10%	53.90%	**57.65%**
	1	5	89.86%	92.66%	**93.45%**	80.89%	81.63%	**84.05%**	53.43%	55.09%	**56.20%**
	5	1	90.22%	92.76%	**93.29%**	80.91%	77.91%	**84.14%**	52.31%	54.82%	**57.99%**
	5	5	90.76%	93.05%	**93.70%**	84.24%	82.71%	**84.64%**	53.88%	55.51%	**56.27%**

TABLE IV: Number of communication round required to reach a certain threshold (Acc. thresh.) for $k_2 = 2$ and variations of k_1 and k_e in IID and non-IID settings. We show the average improvement of CHESSFL compared to both baselines, reaching up to 4.13× faster convergence. Overall, CHESSFL converges faster in both IID and non-IID settings, for all datasets.

Dataset			SVHN (Acc. thresh. 85%)				CIFAR10 (Acc. thresh. 70%)				CIFAR100 (Acc. thresh. 45%)			
	k_1	k_e	SemiFL	FixMatch	**CHESSFL**	**Improv.**	SemiFL	FixMatch	**CHESSFL**	**Improv.**	SemiFL	FixMatch	**CHESSFL**	**Improv.**
IID ($\alpha = 100$)	1	1	224	78	**50**	**3.02**×	194	146	**80**	**2.13**×	176	140	**92**	**1.72**×
	1	5	50	20	**12**	**2.92**×	48	32	**20**	**2**×	50	38	**24**	**1.83**×
	5	1	128	66	**44**	**2.2**×	120	**130**	146	**0.86**×	180	154	**108**	**1.55**×
	5	5	38	16	**10**	**2.7**×	28	26	**16**	**1.69**×	46	38	**22**	**1.91**×
Non-IID ($\alpha = 0.1$)	1	1	336	94	**52**	**4.13**×	302	154	**88**	**2.59**×	196	146	**92**	**1.86**×
	1	5	72	18	**12**	**3.75**×	50	36	**20**	**2.15**×	48	40	**24**	**1.83**×
	5	1	154	74	**48**	**2.38**×	182	180	**140**	**1.29**×	194	148	**110**	**1.55**×
	5	5	44	18	**10**	**3.1** ×	42	28	**20**	**1.75**×	46	36	**22**	**1.86**×

TABLE V: Wall clock time elapsed and energy consumption to reach 70% accuracy on CIFAR10 using IID setting for $k_1 = 1$, $k_e = 1$ and $k_2 = 2$ on NVIDIA Jetson Orin Nano 8GB. We observe CHESSFL reaches 70% test accuracy using 2.61× less total energy and 2.74× less the wall clock time, on average.

	Time per Epoch [s]	Energy per Epoch [J]	Total Energy [J]	Wall Clock Time [s]
SemiFL	8.11	15.41	3,252.35	1,712.16
FixMatch	**2.99**	**6.76**	980.62	433.19
CHESSFL	4.72	9.77	**811.17**	**391.98**
Avg. Improv.	**1.18**×	**1.13**×	**2.61**×	**2.74**×

C. Ablation Studies - Discussing Endgames

Tactical Tweaks: We show in Table III and Table IV the impact of varying k_1 and k_e on accuracy and on communication rounds required to reach a certain threshold accuracy, respectively. The main observation is that using $k_e = 5$ boosts the convergence rate for all approaches significantly, by up to 9× compared to $k_e = 1$. This shows how important training on the edge servers is in order to help the learning of unlabeled data on all clients. As mentioned before, we find that training more often on the edge servers makes HFL much more efficient. We depict in Fig. 5 the global test accuracy for CIFAR100 in both IID and non-IID settings. As can be seen, varying k_1 and k_e has little to no impact on the overall performance and convergence speed of CHESSFL compared to the baselines, in both IID and non-IID settings.

Finally, in Fig. 6 we motivate the need for SSHFL by showing the impact of training only on the limited available labeled data. For this experiment we use CIFAR10 IID using

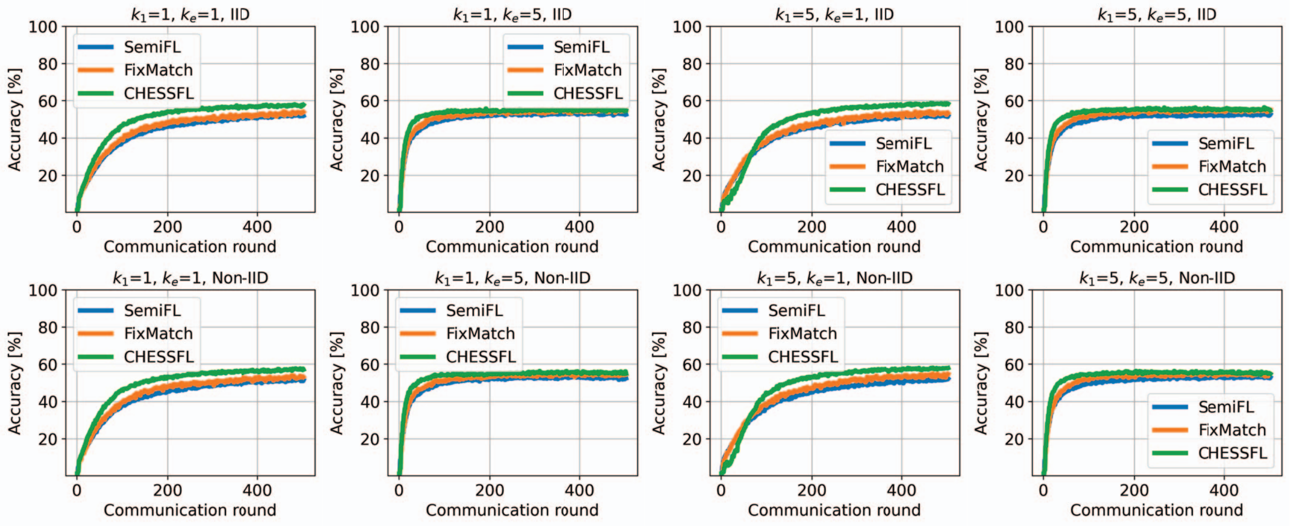

Fig. 5: Global test accuracy results for $k_2 = 2$ and variations of k_1 and k_e for CIFAR100 dataset. CHESSFL achieves higher accuracy in all scenarios when compared to the baselines, for both IID and non-IID settings.

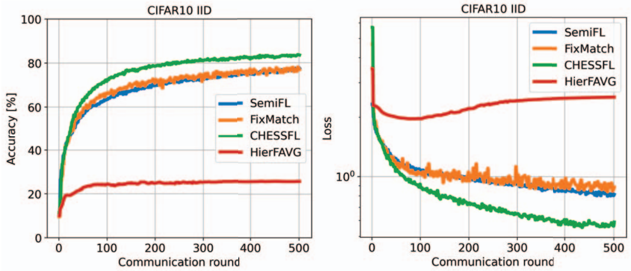

Fig. 6: Global test accuracy and test loss results using IID setting for $k_1 = 1$, $k_e = 1$ and $k_2 = 2$. Using 4,000 labeled images with HierFAVG does not converge, with an accuracy gap of over 50%, thus motivating the need for SSHFL.

$k_1 = 1$, $k_e = 1$ and $k_2 = 2$. We use the 4,000 labeled images split IID between 50 clients with HierFAVG and compare it against using 4,000 labeled images at the edge servers and 50 clients with 500 unlabeled images per client, split IID for SemiFL, FixMatch and CHESSFL. As shown in Fig. 6, HierFAVG is not converging and even shows signs of overfitting. With an accuracy gap of over 50% in global test accuracy, we cannot properly learn in HFL with limited labeled data. Therefore, we need to leverage unlabeled data using semi-supervised HFL.

Efficient Endgame: In Table V we evaluate on a NVIDIA Orin Jetson Nano with 8GB RAM the wall clock time elapsed and total energy consumption required to reach 70% accuracy on CIFAR10 IID. For SemiFL, FixMatch and CHESSFL we run the experiment using $k_1 = 1$, $k_e = 1$ and $k_2 = 2$. If we assume 500 images and 100% pseudo-labeling accuracy, FixMatch and CHESSFL would use all 500 images, while SemiFL would use 1,000 images, *i.e.,* 500 for the fixed dataset and 500 for the mixed dataset. As shown in Table V, the time for pseudo-labeling and constructing the mixed dataset makes the SemiFL approach have up to 72% longer time per epoch. Even considering that FixMatch has up to 58% less local computation compared to CHESSFL, the convergence rate of CHESSFL showed in Table II and Table IV improves the total wall clock time required to reach a given accuracy threshold compared to FixMatch, as shown in Table V. In addition to its computational efficiency, CHESSFL also has negligible communication overhead. Specifically, an additional 1% of the edge model size is required for sending the embedding centroids to the clients. This is valid for most modern model architectures, since the feature embeddings will always be significantly smaller compared to the actual model size.

V. Conclusion: Closing the Game

We have proposed Clustering Hierarchical Embeddings for Semi-Supervised Federated Learning (CHESSFL). To the best of our knowledge, this is the first semi-supervised solution for HFL which considers hierarchical embeddings shared between the clients, edge servers and the cloud. We proposed the Sicilian Defense Loss (SDLoss) to train the edge servers on labeled data and enhance their capability to have robust and diverse feature embeddings. We also introduced the Pawn Promotion Loss (PawnLoss), to facilitate the convergence of client embeddings toward the embedding centroids received from the edge server. Compared to state-of-the-art semi-supervised federated learning approaches, CHESSFL converges up to 5.11× faster and achieves higher accuracy on SVHN, CIFAR10 and CIFAR100 datasets for both IID and non-IID settings. The SDLoss and PawnLoss enable CHESSFL to be more resilient to non-IID data. CHESSFL achieves this performance with vastly improved wall clock time for reaching a certain accuracy threshold and negligible communication overhead.

Future work: Our method can be improved in several ways, *e.g.,* by adding differential privacy to all communication channels and by enabling different models to be used by users with different hardware capabilities. Using labeled and unlabeled data, new attacks can be devised throughout the client-edge-cloud hierarchy, hence the need for new adversarial solutions. All these ideas are left for future work.

Acknowledgment

This research was supported in part by NSF Grant CCF-2107085 and in part by Cisco Research, Inc.

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*, pp. 1273–1282, PMLR, 2017.

[2] S. Sinha, "State of iot 2023: Number of connected iot devices growing 16% to 16.7 billion globally," 2023.

[3] 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)*, pp. 1–6, IEEE, 2020.

[4] A. Wainakh, A. S. Guinea, T. Grube, and M. Mühlhäuser, "Enhancing privacy via hierarchical federated learning," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 344–347, IEEE, 2020.

[5] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?," *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.

[6] V. Chandrasekaran, S. Banerjee, D. Perino, and N. Kourtellis, "Hierarchical federated learning with privacy," *arXiv preprint arXiv:2206.05209*, 2022.

[7] Y. Jin, Y. Liu, K. Chen, and Q. Yang, "Federated learning without full labels: A survey," *arXiv preprint arXiv:2303.14453*, 2023.

[8] T. E. Parliament, "General data protection regulation," 4 2016.

[9] S. of California Department of Justice, "California consumer privacy act," 2018.

[10] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," in *International Conference on Learning Representations*, 2021.

[11] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[12] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[13] P. V. Tran, "Exploring Self-Supervised Regularization for Supervised and Semi-Supervised Learning," in *NeurIPS Workshop on Learning with Rich Experience: Integration of Learning Paradigms*, 2019.

[14] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8866–8870, IEEE, 2020.

[15] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang, "Hierarchical federated learning through lan-wan orchestration," *arXiv preprint arXiv:2010.11612*, 2020.

[16] C. Feng, H. H. Yang, D. Hu, Z. Zhao, T. Q. Quek, and G. Min, "Mobility-aware cluster federated learning in hierarchical wireless networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8441–8458, 2022.

[17] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, p. 896, Atlanta, 2013.

[18] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in neural information processing systems*, vol. 33, pp. 596–608, 2020.

[19] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18408–18419, 2021.

[20] Y. Chen, X. Tan, B. Zhao, Z. Chen, R. Song, J. Liang, and X. Lu, "Boosting semi-supervised learning by exploiting all unlabeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7548–7557, 2023.

[21] Y. Jiang, X. Li, Y. Chen, Y. He, Q. Xu, Z. Yang, X. Cao, and Q. Huang, "Maxmatch: Semi-supervised learning with worst-case consistency," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5970–5987, 2022.

[22] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[23] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1476–1485, 2019.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[25] T. Han, J. Gao, Y. Yuan, and Q. Wang, "Unsupervised semantic aggregation and deformable template matching for semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9972–9982, 2020.

[26] E. Diao, J. Ding, and V. Tarokh, "Semifl: Semi-supervised federated learning for unlabeled clients with alternate training," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 17871–17884, Curran Associates, Inc., 2022.

[27] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[28] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[29] E. S. Lubana, C. I. Tang, F. Kawsar, R. P. Dick, and A. Mathur, "Orchestra: Unsupervised federated learning via globally consistent clustering," *arXiv preprint arXiv:2205.11506*, 2022.

[30] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[31] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

[32] J. Z. HaoChen, C. Wei, A. Gaidon, and T. Ma, "Provable guarantees for self-supervised deep learning with spectral contrastive loss," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5000–5011, 2021.

[33] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[34] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[35] 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.

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[37] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.