Ensuring Trustworthiness in Decentralized Systems through Federated Distillation and Feature Mixing

Christos Chatzikonstantinou[1], Athanasios Psaltis[1,2], Charalampos Z. Patrikakis[2], and Petros Daras[1]

[1]Centre for Research and Technology Hellas, Thessaloniki, Greece
[2]Dept. of Electrical and Electronics Engineering, University of West Attica, Athens, Greece
{chatziko, at.psaltis, daras}@iti.gr {apsaltis, bpatr}@uniwa.gr

***Abstract*—In this work, a novel federated distillation weight aggregation method is proposed. Specifically, an algorithm designed for effective learning in distributed environments is introduced. This algorithm includes an innovative federated distillation scheme, proposing a sophisticated aggregation of model outputs, employing a global server model to manage process. On the server side, feature mixing is employed to aggregate client representations before they are transmitted back to the client side for knowledge distillation. During feature mixing a weight factor is assigned to each client's logits, penalizing bad quality clients and preserving system's credibility. Thorough experimentation has been conducted to comprehensively study the issue at hand. Key findings reveal the significant potential of the proposed solution, which achieves robust performance in a federated setting while reducing communication costs.**

***Index Terms*—federated distillation, feature mix, representation learning, image classification**

I. Introduction

The exponential growth in the use of smart devices precipitates a marked increase in data transmission volumes, presenting a significant challenge in the field of distributed data management. A paramount concern in this domain is the trustworthiness of those systems, as well as data privacy and protection. To overcome this challenge, Federated learning (FL) [1] has been developed as a pivotal methodology, enabling the training of a centralized model whilst ensuring the retention of users' sensitive data exclusively within their personal devices. Moreover, FL can be more robust to poisoning attacks aiming to harm a system's trustworthiness since the model outcomes depend on the contribution of multiple models, whereas in centralized systems a malicious attack to the training model can cause a severe effect to system's reliability.

Nevertheless, the FL approach involves a communication overhead being proportional to model sizes. Particularly, models of large size prove to be impractical for deployment. Furthermore, the necessity for model aggregation in the FL training process imposes a lack of flexibility in the architecture of client models, as each client is required to train a model with an identical architectural framework.

Towards this direction, a novel technique known as Federated Distillation (FD) has been put forward. This approach diverges from the traditional model aggregation method, opting instead for the exchange and aggregation of client model logits. These aggregated logits are then utilized in the model distillation phase to disseminate the knowledge acquired by the local datasets among all clients. This alternative methodology offers advantages such as reduced communication costs, greater flexibility in client model architecture, and improved handling of non-IID (non-independently and identically distributed) data. At the same time, data are not exchanged between users and the central server and the system remains robust against poisoning attacks.

The majority of FD methods in the literature, predominantly utilize averaging of client logits as their aggregation technique. However, there has been limited exploration of alternatives to this approach. The proposed technique delves into the client logit fusion process during the aggregation process, which we contend is a crucial factor affecting the performance of the global model. Consequently, this paper introduces an innovative aggregation method designed to refine the selection criterion.

During the client output aggregation, label change can occur. In contrast, feature mixing can mitigate label change by reducing the weight factor (and the impact on aggregated output) of clients that are mostly "responsible" for label change. In this respect, drawing inspiration from the work of Parvaneh *et al*. [2], a feature mixing aggregation method is proposed, although with a different objective. In [2], the potential label change resulting from feature mixing identifies the images that require labeling. Conversely, in our approach, any potential label change is considered undesirable and leads to a reduction in the weight factor. In this way, the proposed aggregation method can deal with bad quality client logits that may occur either due to bad training or due to a malicious attack to this specific node. A trainable alpha vector for the weighted aggregation of client logits is introduced, while a global model is maintained on the server side to train this vector. The main contributions of this work are as follows:

- The introduction of an innovative federated distillation scheme, which leverages a global server model to guide the aggregation process, encompassing a more advanced aggregation of model outputs.
- The proposal of a novel aggregation method is outlined. This method utilizes feature mixing and incorporates

a trainable vector, designed to assign reduced weight factors to images that are likely to induce label changes.
- The effectiveness of the proposed approach is validated through extensive comparative analysis on three of the most comprehensive publicly accessible benchmarks, demonstrating its superiority and robustness across a diverse array of FL scenarios. Experiments evaluating the system's resilience against poisoning attacks have also been performed.

II. Related Work

A. Knowledge Distillation

Knowledge distillation is a strategy for efficiently transferring knowledge from a large model (the teacher model) to a smaller one (the student model). The teacher model guides the student model that improves performance through iterative learning. Knowledge distillation is widely used for model deployment on resource-limited devices. In [3], the concept of distillation learning was introduced, using soft outputs from the teacher model to guide the training of a smaller model. In addition, a distillation loss and cross entropy loss are utilised to balance between data fitting and teacher imitation. Zhao et al. [4] split knowledge distillation into two parts: target and non-target. The target knowledge distillation component is a binary logit distillation of the target class, whereas the non-target knowledge distillation part is a multi-category logit distillation of non-target classes. In [5] the deviation between teacher and student model prediction is studied. A correlation-based loss is employed to capture both inter-class and intra-class relations from the teacher. Finally, the knowledge distillation technique has been extended to the field of FL, as described in II-B.

B. Federated Distillation

Unlike traditional centralized machine learning (ML) techniques, FL trains an algorithm through numerous independent sessions, each with its own distinct dataset. Initially, research on FL focused on improving communication efficiency and model updates. McMahan *et al*. [1] introduced a novel concept of averaging local stochastic gradient descent updates (known as FedAvg) to improve client information usage during communication rounds. To address challenges such as low device participation and non-IID local data, studies have explored online knowledge distillation methods. FD introduces a novel perspective to federated learning, emphasizing the exchange of model outputs rather than model parameters.

1) Federated Distillation without server model: The FD technique was first introduced by Jeong *et al*. [6], where they proposed using a Generative Adversarial Network (GAN) as a central model. In this approach, clients upload per-label averaged soft targets to train the GAN. Subsequently, clients download the GAN generator and produce samples for underrepresented labels, aiming to achieve an IID dataset. A prevalent approach in the literature involves the use of a public dataset accessible to all clients for local distillation, leading to improved model performance. In [7], Li *et al*. presented a scenario in which each client possesses a small labeled dataset alongside a larger public dataset accessible to all. The training process involves initial training on this communal dataset, followed by training on the private dataset. Logit vectors are then transmitted to the server for aggregation. In their study [8], Itahara *et al*. introduced a semi-supervised method that employs a labeled private dataset and a communal unlabeled dataset. They altered the aggregation step by proposing an Entropy Reduction Aggregation (ERA), demonstrating that using a temperature lower than one when applying softmax to aggregated logits reduces the entropy of global soft targets. This approach is particularly beneficial in non-IID settings.

2) Federated Distillation with server model: A subset of methods employ the server solely as an aggregator (similar to conventional FL methods) for locally computed model logits. More recent strategies have incorporated a server distillation step, which distills a server-side model that can be used to construct global logits (or soft targets) for broadcasting. For instance, Cheng *et al*. [9] employ both a communal dataset and a private dataset, utilizing smaller client models alongside a larger server model. The server categorizes instances in the communal dataset into correctly and incorrectly predicted subsets to enhance convergence. In [10], a one-shot distillation method is introduced. The proposed technique combines distillation and aggregation mechanisms to support FL. In this approach, client models are fully trained prior to being transmitted to the server, which then aggregates per-class attention maps.

Common ground of the existing federated distillation methods that employ a central server model is the lack of assessment of the client logits transmitted. In this work, the server model assesses the quality of the transmitted logits, penalizing the ones that would cause label change, and thereby ensuring the system's trustworthiness.

III. Proposed method

A. Problem statement

In a federated system, data are inherently confined to individual clients, and their exchange with other clients is stringently prohibited. Following a common literature practice in FD systems, each client possesses a distinct local dataset. In addition, there exists a communal dataset, shared among all clients. On the server side, a global model is maintained, which is trained using this communal dataset.

A main challenge in federated distillation systems lies in the aggregation of the client logits on the server side. The prevailing method in the literature is to simply average these logits. However, this approach, fails to consider that data heterogeneity or varying sizes of local datasets might lead some local models to transmit lower-quality logits to the server, which could subsequently decrease local models' performance during distillation process. To address this issue, the proposed method adopts a strategy where client logits are gradually added, with a specific focus on penalizing those logits that result in a label change.

B. Local client training

The local clients undergo supervised training, where each client is assigned a distinct segment of the dataset. This allocation remains consistent throughout all federated rounds, with clients training exclusively on their respective segments. The training subset of the dataset is used for model training purposes, whereas the validation subset serves to assess and verify the performance of the models trained by each client. During the training phase, the Cross-Entropy loss function is utilized:

$$L_{ce}(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} y_{mk}\log p_{mk} \quad (1)$$

where θ represents the model parameters, M denotes the total number of images, K the total number of classes, y_{mk} the label of the m,k image and p_{mk} the class probability of the m,k image.

Aside from each client's training and validation sets, there is a communal dataset accessible to all clients, following the common literature practice ([7], [9]). This dataset acts as a reference point for addressing the issue of data heterogeneity across clients. Each client produces model outputs for each sample in the communal dataset at the end of each federated round. These outputs are then transmitted to the server for aggregation, creating global anchors.

After the aggregation that is performed on the server side, the global anchors are transmitted back to the clients. To ensure knowledge alignment across all clients, in the subsequent federated round, each client engages in knowledge distillation using the aggregated outputs derived from the communal dataset (the pipeline is also illustrated in Figure 2). The knowledge distillation term is computed using the following expression:

$$\mathcal{L}_{KL} = ||\Phi(o_r^c) - \Phi(o_{r-1})|| \quad (2)$$

where Φ represents the network function, o_r^c signifies the model output of client c at the federated round r and o_{r-1} denotes the global anchors from the previous round. The addition of this term to the total loss distills the knowledge of the aggregated model outputs to each local model. Contrary to the majority of the literature methods that employ global anchors as soft labels, in this work the mean squared error is used, because it achieves a superior performance.

C. Multi-scale knowledge distillation

The limited amount of data that is communicated from the clients to the data in the federated distillation scheme is a disadvantage, as only the output from the model's last fully connected layer is sent to the central server. To mitigate this limitation, a multi-scale knowledge distillation strategy is implemented, drawing inspiration from the research of [11] (Figure 2). Specifically, each client transmits certain intermediate representations (also known as hidden representations), computed over the communal dataset, in addition to the model's output; the number and the position of these intermediate representations can vary depending on the specific requirements of the problem.

Different-level features can be extracted, let $z_l^N = f_l^N(\cdot), l \in (1,...,L)$, where l are the different model layers, z_l^N is the intermediate representation at layer l for the client N, with size $H*W*C$ and $f_l^N(\cdot)$ is the subnetwork for feature extraction up to the layer l. Then, the final form of the feature map can be obtained by concatenating the $W*C$ height-pooled slices and the $H*C$ width-pooled slices for h_l^N:

$$\Phi(z_{l_{j-1}}^N) = \left[\frac{1}{W}\sum_{w=1}^{W} z_{l_{j-1}}^N[:,w,:] \,\middle|\middle|\, \frac{1}{W}\sum_{h=1}^{H} z_{l_{j-1}}^N[h,:,:]\right] \quad (3)$$

where $[\cdot||\cdot]$ denotes concatenation over the channel axis, N is the number of clients, j is the federated round and $\Phi(\cdot)$ is the network function. The intermediate representations are calculated over the communal dataset, therefore all clients extract features from the same subset. After all intermediate representations of all clients are extracted, they are averaged per client, $\Phi(z_l) = \frac{1}{N}\sum_{n=1}^{N}\Phi(z_{l_{r-1}}^N)$. At the beginning of the next federated round, for each client, the Euclidean distance between the mean averaged representation of the clients and each client's local representation is calculated. The multi-scale knowledge distillation term is calculated as:

$$\mathcal{L}_{multi-scale_{KL}} = \frac{1}{L}\sum_{l=1}^{L}\left|\left|\Phi(z_{l_j}^N) - \Phi(z_{l_{j-1}})\right|\right| \quad (4)$$

In this work, ResNet56 model [12] is utilized on client side, therefore three intermediate representations are utilized, corresponding to the outputs of three ResNet layers.

The final form of the loss function is as follows:

$$L_f = L_{ce} + m_c * \mathcal{L}_{multi-scale_{KL}} + L_{KL} \quad (5)$$

where m_c is the coefficient of the multi-scale knowledge distillation term.

D. Global representation aggregation

In this work a feature mixing aggregation method is proposed, inspired by [2], i.e., the linear interpolation of different client outputs.

More specifically, for each client, a tensor a of random numbers is generated from a normal distribution, with mean and standard deviation equal to a hyperparameter denoted as a_{cap}. The dimension of the tensor is equal to the model output, calculated as $n_{classes} * n_{samples}$, where $n_{classes}$ represents the number of the dataset classes and $n_{samples}$ represents the samples' number of the communal dataset. This tensor is utilized to perform interpolation between the current client's model output and a convex combination of the model outputs from the previous clients.

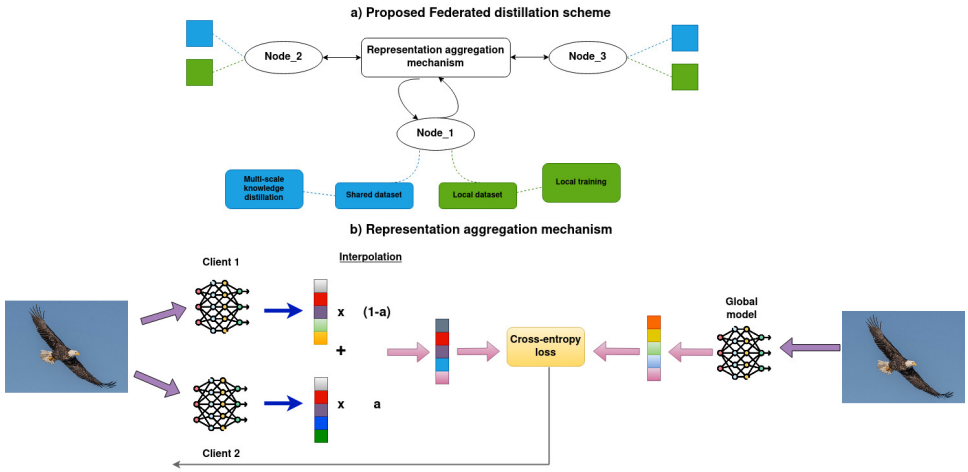

Fig. 1. (a) The outline of the proposed algorithm: Each node is trained with its local dataset and a multi-scale knowledge distillation strategy is implemented (inspired by [11]), employing the communal dataset. Model outputs are transmitted to the server and they aggregated using the proposed representation aggregation mechanism. (b) The representation aggregation mechanism on the server-side: The linear interpolation between two clients (or a client and a convex combination of the previous ones) is computed using the equation 6 and along with the global model output are utilized to calculate the learnable a tensor cross-entropy loss.

The interpolation process begins with the first client's model output being combined with the second client's model output. Subsequently, the model output of the third client is interpolated with the convex combination of the model outputs from the first two clients, and so on. The interpolation method described above has also been experimentally evaluated in Section IV-C2 Linear interpolation is computed using the following equation:

$$O_{fm} = (1 - a)\, o^* + a\, o^n \tag{6}$$

where a represents the generated tensor, o^* signifies the convex combination of the clients' model outputs (or the first client) and o^n denotes the model output of the client $n, n \in [0, n_{clients}]$, $n_{clients}$ represents the number of clients.

On the server side, there exists a server model that undergoes training using the communal dataset. The server model output is then compared with O_{fm}. The Cross-Entropy loss function is utilized as the loss function of the global model:

$$L_G(\theta) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} y_{mk} \log p_{mk} \tag{7}$$

The tensor a is learnable and its loss is calculated as follows:

$$L(a) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} o_{g_{mk}} \log(O_{fm_{mk}}) + c_n norm(a) \tag{8}$$

where $o_{g_{mk}}$ is the prediction of the global model of the m, k image (treated as the respective label), $O_{fm_{mk}}$ is the linear interpolation of the m, k image, the norm refers to the Frobenius norm of the tensor and c_n is the coefficient of the norm. After training the a tensor and prior to the feature mixing step, it is limited to the range $[0, 1)$. Thus, the Frobenius norm penalizes large values of a.

The process outlined in this section is repeated $n_{clients} - 1$ times to integrate each client's model output for a specified number of epochs. To prevent overfitting of the global model, inspired by [13], the global model's gradients are scaled by $1 \oslash O_{diff}$, where $\oslash$ denotes the element-wise division, as indicated below:

$$(\frac{\partial L_G}{\partial W})' = 1 \oslash O_{diff} \otimes \frac{\partial L_G}{\partial W}, \tag{9}$$

where W represents the model's parameters and $\otimes$ denotes the element-wise multiplication. O_{diff} is a hyperparameter whose value was selected based on the experiments presented in IV-C3

E. FedFMRL Algorithm

This section presents the training flow of the FedFMRL procedure, which is organized into four main algorithms. In Algorithm 1, the main outline of the proposed method is provided. Algorithm 2 describes the training and update steps within each local client, while Algorithm 3 details the distillation process carried out by the local clients. Lastly, in Algorithm 4, the methodology for aggregating global representation is presented.

IV. Experimental Results

A. Dataset settings

The experimental results of the proposed FedFMRL system, utilizing the CIFAR-10, CIFAR-100 [14] and TinyImagenet [15] image classification datasets as a benchmark, are reported in this section. These datasets were adapted to simulate both IID and non-IID scenarios, incorporating various FL parameters. In the IID setting, the data was balanced, ensuring an even distribution among clients. Conversely, in the non-IID setup, we deliberately created a scenario with extreme data imbalance. Similarly to prior works [16], a concentration

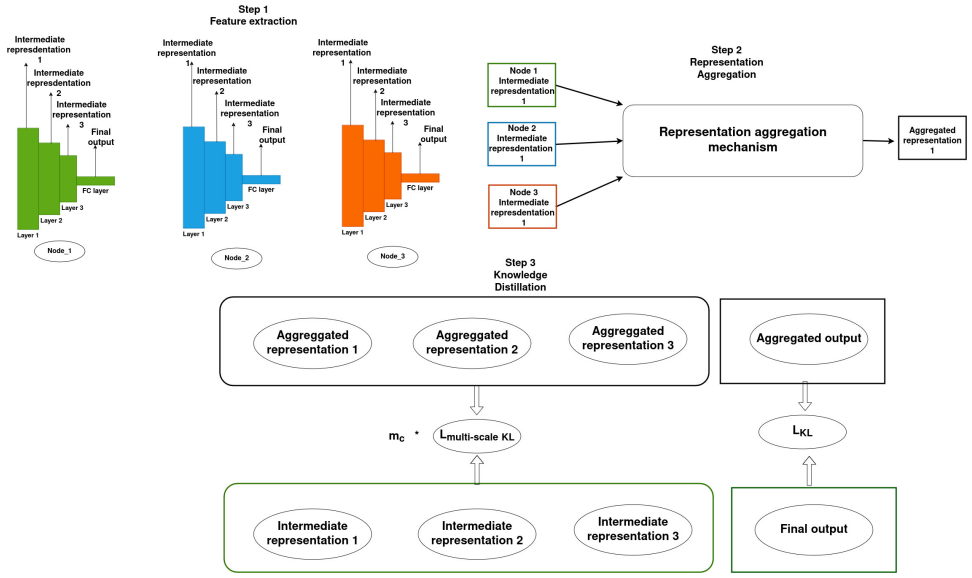

Fig. 2. The pipeline of the knowledge distillation is illustrated in the first diagram. Initially, client nodes are extracting their intermediate representations and the final output, computed over the communal dataset. Then, the intermediate representations and the final output are aggregated on the server side, employing the representation aggregation mechanism. At the beginning of the next federated round, knowledge distillation step is performed, calculating the multi-scale knowledge distillation term (added to the final loss with the m_c coefficient) and the knowledge distillation term.

Algorithm 1 FD Algorithm

Require: T is the number of communication rounds, $\mathcal{N}$ is the total number of clients, θ_l^i represents the parameters of the local models, θ_g represents the parameters of the global model, $\mathcal{D}_l^i$ are the separate datasets for the local clients, $\mathcal{D}_{cm}$ is the communal training dataset, $\mathcal{D}_{test}$ is the communal testing dataset, e_g are the epochs of the global model, η is the learning rate and a_η is the learning rate of the tensor a.

for each i from 1 to N **do**
  Initialize local models θ_{l_i}
  Prepare local datasets D_{l_i}
end for
Initialize global models θ_g
initialize tensor a
Prepare communal training dataset D_{cm}
Prepare communal testing dataset D_{test}
Server executes:
for each Federated round $j = 0, 1, 2, \ldots$ **do**
  for each client in parallel **do**
    $\theta_{l_j} \leftarrow$ ClientUpdate(D_{l_j}, θ_{l_j})
    $z_j, .. \leftarrow$ Extract representations from (θ_{l_j})
    Send representation to server
  end for
  $a_j \leftarrow$ AnchorAggregation$(D_{cm}, \theta_g, z_j, N, a, e_g, a_\eta)$
  Distribute the updated global anchors
  for each client in parallel **do**
    $\theta_{l_{j+1}} \leftarrow$ KDUpdate$(D_{cm}, \theta_{l_j}, a_j)$
  end for
end for
Validate the updated distillated models on D_{test}

Algorithm 2 ClientUpdate Function

ClientUpdate(D_l, θ): ▷ Run on specific client
for each local epoch i from 1 to E **do**
  for each batch in D_l **do**
    $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l, \text{labels})$ ▷ Update the client model with Eq. 1
  end for
end for
return θ_l

Algorithm 3 DistillationUpdate Function

KDUpdate(D_{cm}, θ_l, a): ▷ Run on specific client
for each batch in D_{ex} **do**
  l_{mul} ▷ Compute Multi Scale loss with Eq. 4
  l_{dis} ▷ Compute Distillation loss with Eq. 2
  $L = m_c * l_{mul} + l_{dis}$
  $\theta_l \leftarrow \theta_l - \eta \nabla L(\theta_l)$ ▷ Update the client model with aggregated loss L
end for
return θ_l

parameter $beta$ is used to produce the non-IID data partition among clients. A total number of 10 participating nodes were involved, with images distributed among them for experimentation.

B. Implementation Details

1) Architecture and Parameters: The architecture of the proposed method is grounded in a distributed structure, employing the ResNet56 [12] model, for the local clients and the ResNet50 model, on the server side. Local clients un-

Algorithm 4 AnchorAggregation Function

AnchorAggregation$(D_{cm}, \theta_g, z_j, N, a, e_g, a_\eta)$:
for each i from 1 to $N-1$ **do**
  for each j from 1 to e_g **do**
    $\theta_g, o_g \leftarrow$ ClientUpdate(D_{ex}, θ_g)
    O_{fm} ▷ Compute linear interpolation with equation 6
    $a \leftarrow a - a_\eta \nabla L(a, o_g, O_{fm})$ ▷ Update the a tensor with Eq. 8
  end for
end for
return a

dergo training using a multi-loss approach introduced in [11], which involves three intermediate outputs corresponding to the outputs of three ResNet layers. The SGD [17] technique is employed as the optimization technique. The learning rate is set to 0.1, with a weight decay of $1e-3$, c_n is 0.01 and m_c is equal to 0.1. To ensure comprehensive learning and convergence, the local models are trained for 300 epochs.

The server model is trained for 5 global epochs on the communal dataset for each feature mixing step. The learning rate is set to 0.1, with a weight decay of $1e-5$, and c_n equal to 0.01. Regarding the learning rate of the a tensor, it is 0.1 for the first $\frac{2}{3}$ of the global epochs and then increased to 10 for the remaining training process.

Deep learning models are implemented in Python 3.8 within the PyTorch (version 2.0) [18] environment. The code is available at: https://github.com/chatzikon/FEDFMRL

2) Federated setting: The training paradigm for the proposed FL system, comprising a central server and 10 local clients, encopasses 6 federated rounds. In each round, local clients are separately trained for 50 epochs, with no inter-client communication during training. The final evaluation of the local models is conducted using the test set of the CIFAR-10, CIFAR-100 and Tiny ImageNet datasets. The training set is divided into a communal set that is accessible to all clients, constituting 20% of the training set, and the remaining 80% is distributed among the clients. Each client retains 10% of its local data as its local validation set and uses the rest for its local training set.

3) Baselines: To comprehensively assess the proposed method's efficacy, a comparative performance analysis was conducted using two distinct configurations. Initially, benchmark tests were carried out, encompassing both IID and non-IID scenarios. Furthermore, the proposed method was compared with a scheme of locally isolated clients (with no communication between them), a standard federated distillation strategy and a variation of the proposed method, without the use of the multi-scale knowledge distillation term (Eq. 4). In order to highlight the distinctive properties of our design, we performed direct comparisons with the aforementioned baseline approaches, as shown in Table VII.

C. Performance Evaluation

This section provides a summary of the results obtained through the application of the proposed scheme in several distinct scenarios under various learning settings. The CIFAR-10 dataset is employed on those scenarios, apart from the non-iid and data poisoning scenarios. In those cases, CIFAR-100 dataset is utilized due to the fact that it includes more classes, allowing for a more reliable evaluation of the proposed method for data poisoning and client data imbalance experiments. Each experiment is repeated 3 times and the mean and standard deviation of the results are presented.

1) Alpha calculation methods: The a tensor can be computed using a closed-form solution or by drawing values from a Gaussian distribution with a predefined mean and standard deviation for the tensor a. The closed-form solution introduced in [2] is employed in this regard. Alternately, in the case of Gaussian-generated a, the tensor can be employed without modifications, or the method described in Section 3.3 for a learnable a tensor can be utilized.

As demonstrated in Table I, a learnable a strategy proved to be more effective in capturing the diverse properties of each client and assigning the appropriate weight factor to each client. This strategy has been adopted for the remaining experiments.

TABLE I
EXPERIMENTS WITH DIFFERENT METHODS FOR ALPHA CALCULATION ON THE CIFAR-10 DATASET. THE CLOSED-FORM SOLUTION IS INTRODUCED IN [2]. THE ALPHA CAN ALSO BE GAUSSIAN GENERATED AND LEARNABLE (SECTION 3.3).

Alpha calculation	Mean Client Accuracy	Global Model Accuracy
Closed-form	60.11 ± 0.68	N/A
Gaussian generated	60.22 ± 0.68	N/A
Gaussian generated with learnable a	$\mathbf{60.59 \pm 0.34}$	$\mathbf{48.41 \pm 0.34}$

2) Global aggregation schemes: In this section, the impact of various global aggregation schemes is evaluated. Three different methods are subjected to testing:

- **Scheme A**: This method involves weighted averaging of the client outputs. For each client, the factor $(1-a) * o_r^c$ (representing the model output of client c at the federated round r) is added to the soft logits tensor. The final soft logits tensor is then divided by the product of the number of clients and the a_{cap}
- **Scheme B**: In this scheme, a linear interpolation is computed, but the o^* of the Eq. 4 defines the mean model output of all clients except client n. This is followed by division by the number of clients.
- **Scheme C**: This method follows the process described in Section 3.3.

The results (as presented in Table II) demonstrate that linear interpolation is a better method for representation aggregation compared to weighted averaging. The scheme C achieves better results in both the client and the server sides and has

been chosen as the aggregation scheme for the subsequent experiments.

TABLE II
COMPARATIVE EVALUATION OF DIFFERENT GLOBAL AGGREGATION SCHEMES ON THE CIFAR-10 DATASET. SCHEME A INVOLVES WEIGHTED AVERAGING OF THE CLIENT OUTPUTS. IN SCHEME B A LINEAR INTERPOLATION IS COMPUTED. THE SCHEME C IS THE PROPOSED METHOD.

Scheme	Mean Client Accuracy	Global Model Accuracy
A	10.00 ± 0.00	48.39 ± 1.02
B	59.93 ± 0.99	48.39 ± 1.02
C	$\mathbf{60.59 \pm 0.34}$	$\mathbf{48.41 \pm 1.08}$

3) Impact of the O_{diff} factor: In this section, the impact of different values of the O_{diff} factor on the accuracy of the server's model and the clients' models is examined. According to Table III the global model attains the highest accuracy when O_{diff} is set to 2. Thus, this value is employed in the subsequent experiments. It is observed that the client models are not significantly affected by the variation in the O_{diff} factor. This outcome is expected because the global model is directly influenced during training by O_{diff}, whereas the client models are indirectly affected due to the output modification of the global model.

4) Impact of the a_{cap} factor: This experiment addresses the effect of different values of a_{cap} on the local models' accuracy. The a_{cap} value determines the mean and standard deviation of the Gaussian distribution used to generate the initial a values. As illustrated in Table IV, the optimal performance is achieved when $a_{cap} = 0.2$. Therefore, this value has been selected for use in subsequent experiments.

5) Impact of client number: In this subsection experiments with varying number of clients are presented. Those experiments show a decline in local accuracy but stable global accuracy (illustrated in Table V). Increasing clients decreases local accuracy, as expected, but global accuracy remains stable up to 40 clients, demonstrating the method's robustness.

6) Impact of communal dataset size: The study on the size of the communal dataset aims at specifying the size that optimizes the client accuracy. Moreover, for this and the subsequent experiments the knowledge distillation term is introduced (Equation 4). As illustrated in VI, the global model achieves a better performance, the larger the communal data set, as expected. However, a larger communal dataset leads to fewer data to be shared among clients. To ensure that the client models accurately capture the unique patterns of each client a trade-off is required between the communal dataset size and the client private datasets. Based on the results presented in this section, the communal dataset size was defined to be 20%.

7) Comparison with baseline methods: In this section, three baseline methods are compared with FedFMRL, as described in Section IV-B3, using the CIFAR-10, CIFAR-100 and TinyImagenet datasets. In particular, on the CIFAR-10 dataset, FedFMRL achieves remarkable results with a Mean Client accuracy of 70.72%, surpassing the other methods, accompanied by a Global Model accuracy of 48.41%. Similarly, on the CIFAR-100 dataset, FedFMRL outperforms the other methods with a Mean Client accuracy of 37.87% and a Global Model accuracy of 17.83%. In the case of TinyImagenet, the proposed method achieves a lower accuracy of 19.43 and a Global Model accuracy of 11.38. The degradation of the mean client accuracy is expected because TinyImagenet is a more challenging dataset, with 200 classes, double compared to the CIFAR-100. It is worth noting that FedFMRL achieves superior results compared to the traditional FLD boosted with the knowledge distillation term, highlighting the added value

TABLE III
IMPACT OF DIFFERENT VALUES OF THE O_{diff} FACTOR ON THE CIFAR-10 DATASET.

O_{diff}	Mean Client Accuracy	Global model accuracy
1	59.81 ± 1.20	45.15 ± 0.81
2	60.34 ± 0.44	$\mathbf{48.41 \pm 1.08}$
3	59.89 ± 0.42	46.77 ± 0.52
4	60.42 ± 0.52	47.03 ± 1.38
5	60.58 ± 1.11	46.54 ± 0.94
10	60.09 ± 0.85	41.72 ± 0.25
15	60.32 ± 1.11	40.55 ± 0.56
20	60.16 ± 0.42	39.52 ± 0.44

TABLE IV
IMPACT OF DIFFERENT VALUES OF THE a_{cap} FACTOR ON THE CIFAR-10 DATASET.

a_{cap}	Mean Client Accuracy
0.1	60.47 ± 1.24
0.2	59.62 ± 0.69
0.2	$\mathbf{60.59 \pm 0.34}$
0.4	59.55 ± 1.35
0.5	60.34 ± 0.44
0.6	59.35 ± 0.85
0.7	60.04 ± 0.24
0.8	59.07 ± 0.67
0.9	58.88 ± 1.50

TABLE V
IMPACT OF CLIENT NUMBER

Client number	Mean Client Accuracy	Global model accuracy
5	76.58 ± 0.09	46.61 ± 0.52
10	60.59 ± 0.34	48.41 ± 1.08
20	53.35 ± 1.32	46.72 ± 1.42
30	44.94 ± 1.49	48.03 ± 1.92
40	37.04 ± 1.21	47.33 ± 0.51
50	35.45 ± 0.41	44.12 ± 3.03

TABLE VI
IMPACT OF COMMUNAL DATASET SIZES EVALUATED ON THE CIFAR-10 DATASET.

Communal dataset size	Mean Client Accuracy	Global model accuracy
0.1	42.47 ± 2.98	42.01 ± 0.85
0.2	$\mathbf{70.72 \pm 0.44}$	$\mathbf{48.41 \pm 1.08}$
0.3	65.00 ± 1.47	50.79 ± 1.20
0.4	57.44 ± 2.68	52.12 ± 1.44
0.5	52.78 ± 2.86	57.08 ± 2.51

TABLE VII
COMPARATIVE EVALUATION WITH BASELINE METHODS ON CIFAR-10 CIFAR-100 AND TINYIMAGENET DATASETS

Method	Mean Client Accuracy	Global Model Accuracy
CIFAR-10		
Local Isolated Clients	57.47 ± 0.42	N/A
FLD with multiloss	70.02 ± 0.47	N/A
FedFMRL w/o multiloss	60.59 ± 0.34	48.41 ± 1.08
FedFMRL w multiloss	$\mathbf{70.72 \pm 0.44}$	$\mathbf{48.41 \pm 1.08}$
CIFAR-100		
Local Isolated Clients	23.29 ± 0.26	N/A
FLD with multiloss	37.22 ± 0.56	N/A
FedFMRL w/o multiloss	31.95 ± 0.25	17.83 ± 1.78
FedFMRL with multiloss	$\mathbf{37.87 \pm 0.41}$	$\mathbf{17.83 \pm 1.78}$
TinyImagenet		
Local Isolated Clients	11.09 ± 0.27	N/A
FLD with multiloss	18.99 ± 0.14	N/A
FedFMRL w/o multiloss	11.40 ± 0.38	11.38 ± 0.01
FedFMRL with multiloss	$\mathbf{19.43 \pm 0.33}$	$\mathbf{11.38 \pm 0.01}$

TABLE VIII
EXPERIMENTS WITH NON-IID DATA, FOR DIFFERENT BETA VALUES ON THE CIFAR-100 DATASET. CONCENTRATION PARAMETER BETA IS USED TO PRODUCE THE NON-IID DATA PARTITION AMONG CLIENTS, SIMILARLY TO PREVIOUS WORKS.

beta	0.25	0.5	0.75	1	IID
FLD	16.78 ± 2.89	22.15 ± 0.23	22.82 ± 0.11	24.92 ± 0.41	30.45 ± 0.47
FedFMRL	$\mathbf{17.04 \pm 1.35}$	$\mathbf{24.29 \pm 1.17}$	$\mathbf{25.49 \pm 1.70}$	$\mathbf{28.90 \pm 1.63}$	$\mathbf{37.87 \pm 0.41}$

of the FEDFMRL method. On the other hand, isolated clients who train separately exhibit the lowest accuracy, which was expected. However, this comparison underscores the value of the proposed approach.

8) Impact of non-IID data: Non-IID data often exhibit a wide range of patterns and variations among nodes, making it challenging to learn a generalized model. Therefore, to assess the impact of non-IID data on the proposed method, the mean client accuracy of the FedFMRL method (incorporating the multi-scale knowledge loss, as in Equation 4) is compared with the baseline federated distillation method for different values of beta, as shown in Table VIII. The results demonstrate that the FedFMRL scheme outperforms the baseline federated distillation method across all beta values, highlighting the added value of the feature mixing and the multi-scale knowledge loss in handling non-IID data.

To evaluate the robustness of the proposed method, a different setting was also tested: The total number of clients is restricted to 80, 60 or 40 classes, with beta equal to 0.5. Each client has a different subset of classes. Moreover, experiments with centralized data were performed for comparison. As presented in Table IX, in federated setting, accuracy is decreased a bit more than 10% when each client has 40 classes, while in centralized setting the accuracy decrease is more than 35%. The results of this section demonstrate the ability of the proposed method to deal with various non-IID settings.

9) Impact of poisoned classes: FL raises numerous security and privacy concerns. Poisoning attacks, for example, can have a substantial impact on client models, while malicious attackers can prevent client models from converging or even manipulating their prediction results. The ability of a federated system to defend against poisoning attacks is an extremely critical and difficult task, determining its trustworthiness and robustness. Therefore, targeted experiments using an intentional data poisoning approach, specifically through label flipping, were conducted in this subsection.

TABLE IX
EXPERIMENTS, USING CIFAR-100 DATASET, WITH RESTRICTED NUMBER OF CLASSES FOR EACH CLIENT EMPLOYING TWO DIFFERENT SETTINGS: THE PROPOSED FD SCHEME, NON-IID DATA, BETA=0.5 AND A CENTRALIZED SCHEME

Number of classes per client	Mean Client Accuracy	Single Model Accuracy
100	24.29 ± 0.23	65.44 ± 0.23
80	20.59 ± 1.35	55.11 ± 1.09
60	17.02 ± 2.63	42.67 ± 1.83
40	13.86 ± 4.21	29.14 ± 2.07

TABLE X
IMPACT OF POISONED CLASSES, FOR VARIOUS NUMBER OF CLIENTS, ON THE CIFAR-100 DATASET.

Poisoned classes \ Poisoned Clients	1	2	3	4	5
3	37.86 ± 0.85	37.48 ± 0.54	37.50 ± 3.42	36.17 ± 1.38	34.88 ± 6.17
10	37.51 ± 0.77	36.51 ± 1.05	35.56 ± 3.95	35.90 ± 2.31	34.10 ± 0.53
20	36.54 ± 1.18	35.62 ± 1.44	33.32 ± 2.12	32.00 ± 3.51	30.98 ± 3.56
30	35.80 ± 1.32	34.49 ± 1.56	31.54 ± 2.67	29.37 ± 3.81	27.88 ± 3.98

More specifically, a number of classes (3 to 30) was selected for each experiment. The labels of each class are randomly flipped to another class's labels. For example, in the training data of the selected nodes, all the 'leopard' labels could be swapped by 'bed' labels.

As depicted in Table X, the proposed method has less than 10% accuracy drop with up to almost $\frac{1}{3}$ of the classes being poisoned, even if half of the clients (5 out of 10) are poisoned. On the contrary, Table XI, on the centralized experiment when 30 classes out of 100 are poisoned, it has an accuracy drop of 20%.

V. CONCLUSION

This work introduces a novel FD weight aggregation method known as FedFMRL, which leverages feature mixing of the model outputs at the server side to enhance the accuracy of local models. FedFMRL offers a comprehensive solution for scenarios where a more communication-efficient method than traditional FL methods, like FedAvg, is required, achieving promising results. In addition, by penalizing bad quality clients during logit aggregation can preserve system's crediblity. Extensive experimentation yields valuable insights into the suggested learning strategy, especially in highly distributed settings. This study demonstrates FedFMRL's capacity to generate robust local representations, effectively replacing the computationally intensive transmission of large models, while maintaining trustworthiness and integrity in FL environments.

ACKNOWLEDGMENT

This research is funded under the H2020 project STARLIGHT ("Sustainable Autonomy and Resilience for LEAs using AI against High priority Threats"), which has received funding from the European Union's Horizon 2020

TABLE XI
IMPACT OF POISONED CLASSES ON A CENTRALIZED SETTING, UTILIZING THE CIFAR-100 DATASET.

Poisoned classes	Accuracy
0	65.44 ± 0.87
3	64.11 ± 1.81
10	59.02 ± 1.88
20	51.89 ± 1.52
30	45.43 ± 1.74

research and innovation programme under grant agreement No 101021797.

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] A. Parvaneh, E. Abbasnejad, D. Teney, G. R. Haffari, A. Van Den Hengel, and J. Q. Shi, "Active learning by feature mixing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12237–12246, 2022.

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

[4] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11953–11962, 2022.

[5] T. Huang, S. You, F. Wang, C. Qian, and C. Xu, "Knowledge distillation from a stronger teacher," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33716–33727, 2022.

[6] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[7] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[8] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 191–205, 2021.

[9] S. Cheng, J. Wu, Y. Xiao, and Y. Liu, "Fedgems: Federated learning of larger server models via selective knowledge fusion," *arXiv preprint arXiv:2110.11027*, 2021.

[10] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje, "Ensemble attention distillation for privacy-preserving federated learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15076–15086, 2021.

[11] A. Psaltis, C. Chatzikonstantinou, C. Z. Patrikakis, and P. Daras, "Fedrcil: Federated knowledge distillation for representation based contrastive incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3463–3472, 2023.

[12] 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*, pp. 770–778, 2016.

[13] C. Chatzikonstantinou, D. Konstantinidis, K. Dimitropoulos, and P. Daras, "Recurrent neural network pruning using dynamical systems and iterative fine-tuning," *Neural Networks*, vol. 143, pp. 475–488, 2021.

[14] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[15] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[16] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International conference on machine learning*, pp. 7252–7261, PMLR, 2019.

[17] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.