EMERGING TECH CONFERENCE – Edge Intelligence

Volume 03, 2024, pages 10-16

Proceedings of Emerging Tech Conference:
Edge Intelligence 2024

Personalization in Distributed tinyML
Applications via Adaptive Clustered Federated Learning

Alexios Filippakopoulos, Dimitris Kastaniotis[1], Giorgos Andrianakos and Christos Theocharatos
Irida Labs, Kastritsiou 4, 26504 Patras, Greece
{afilippak,kastaniotis,andrianakos,htheohar}@iridalabs.com

Abstract

In large-scale non-IID distributed settings, where statistical heterogeneity and task diversity among local distributions is prevalent, and devices want to maximize performance on the tasks they have data for, vanilla federated algorithms frequently fail to converge and generalize across all tasks. This study aims to highlight the need for more sophisticated approaches in such settings, by adopting a clustered and personalized federated learning approach. We empirically justify the choices made when designing the framework, while our findings suggest a significant boost in performance and training stability.

1 Introduction

From smart gadgets to cameras and sensors, IoT devices produce an immense volume of data. Due to the distributed nature of these devices, the data produced is highly decentralized and heterogeneous. Therefore, a need for AI solutions that consider this distributed nature of data, along with the constraints imposed by the IoT setting, while operating in a privacy-preserving manner, becomes increasingly apparent. Federated Learning (FL) is a popular distributed framework that allows multiple edge devices, referred to as clients constituting a network, to collaboratively train a shared global model, capable of inferring from all tasks within that network. Each client trains its model locally, on its own data, sharing only its updated parameters. Training is coordinated by a central entity, referred to as the server, whose role is to broadcast the initial weights within the network, receive the updated parameters of all participants, aggregate the global model (Federated Averaging) and broadcast it within the network (H. Brendan McMahan 2016). This constitutes a complete global training round in a federated setting. As a result, local data remain at the edge, rendering it decentralized, while the parameter-only exchange between clients and the server preserves, to an extent, both user and data privacy. However, real-world applications are defined by statistical heterogeneity among local distributions, diversity in tasks, as well as non-IID network-wide distributions. Due to these inherent problems, FL's convergence, along with the generalization capabilities of the global model, can be severely hindered, resulting in many clients being better off training solely on their local data (Farzin Haddadpour 2019). Therefore, aggregating one general purpose global model may not be sufficient for modeling all clients' distributions in large-scale heterogenous settings.

[1] *Corresponding author

To address these challenges, Personalized Federated Learning (PFL) acknowledges the benefits of training with rather inaccessible data in a privacy-preserving manner via FL and aims to equip each client with a unique and personalized model instance, tailored to its local data and client-specific tasks. The dominant approaches for achieving personalization in a federated setting, include meta learning (Chelsea Finn 2017), multi-task learning (Virginia Smith 2018), model regularization (Filip Hanzely 2020) and model interpolation (Yishay Mansour 2020), which we opt for, where obtaining personalized models is achieved by mixing the global (aggregated) model with each locally updated model. Another approach is Clustered Federated Learning (CFL) (Felix Sattler 2019), that tries to find clients with similar tasks and/or distributions and cluster them into a 'subpopulation' whose members federate explicitly with each other. Throughout this paper, the goal is to showcase the need for more sophisticated federated approaches in large-scale, distributionally heterogeneous, and task-diverse settings, while addressing some common questions that arise when utilizing the forthcoming techniques.

2 Problem Formulation

Let a population of $k \in \{1, \ldots, KK\}$ clients, each maintaining a local distribution D_k, of size $|D_k|$, on domain $\Xi \coloneqq X \times Y$, where $X \in \mathbb{R}^d$ is the input domain and $Y \in \mathbb{R}^T$ is the label domain. We denote as $\bar{D} = \bigcup_{k=1}^{K} D_k$ the network-wide (global distribution), which corresponds to the joint distribution of all K local distributions D_k.

For any model $h \in \mathcal{H}$, the loss function is defined as $\ell: \mathcal{H} \times \Xi \to \mathbb{R}^+$, while the true and empirical risks for said model h at a specific local distribution D_k are denoted as $L_{Dk}(h) = E_{(x.y)\sim D_k}[\ell(h(x), y)]$ and $L_{Dk}(h)$ respectively. Thus, any client kk training its model (h_k) on its local distribution D_k will obtain the updated model $h_k^* = argminL_{Dk}(h_k)$. In a federated setting, where clients train in a collaborative manner as previously described, the aim is to obtain one aggregated global model $\bar{h} * = argmin\, \hat{L}_D \left(\sum_{k=1}^{K} \left(\frac{|D_k|}{|D|} \right) h_k \right)$. Similarly, CFL will construct $\mathbb{C}$ clusters of similar clients before training is initiated. Then, for each cluster $cc \in \mathbb{C}$, comprised of $Kc \subset K$ clients, the cluster-level global distribution will be $\bar{D}_c = \bigcup_{k=1}^{Kc} D_k$ and we will obtain an aggregated global model $\bar{h}_c * = argmin\, \hat{L}_{D_c} \left(\sum_{k=1}^{K_c} \left(\frac{|D_k|}{|D_c|} \right) h_k \right)$.

On the other hand, PFL via model mixing constructs the personalized model of a client k with the following convex combination: $h_k * = \lambda_k^* + (1 - \lambda)\bar{h^*}$, where λ is the mixing parameter and thereby $h_k * = argmin\, \hat{L}_{D_k}(\lambda \mathrm{h}_\mathrm{k}^* + (1 - \lambda)\bar{h^*})$. The primary issue of this approach stems from the mixing of the global model $\bar{h^*}$ with every client model h_k^*. Specifically, the greater the divergence between $\bar{D}$ and D_k, the more likely it is a suboptimal set of parameters for h_k^* and subsequently a worse fit for D_k to be obtained. As a result, CFL with PFL can be combined to reduce the divergence between the global and some local distributions. By constructing similarity clusters and mixing the cluster-level global models with the corresponding client-side trained models, a superior set of parameters for each goal of maximizing personalization. Hence, a mechanism for identifying and grouping similar clients into clusters, whose joint distribution will diverge less from D_k compared to $\bar{D}$, is critical, since the quality of the obtained parameters is intrinsically linked to the quality of the clusters formed.

3 Methodology

To construct the clusters before the commencement of training, Hierarchical Clustering is utilized while assuming no knowledge of a client's data, tasks or distribution. To quantify the similarities of clients, the following intuition is used. In supervised statistical learning, both empirically and theoretically, it is proven that parameters that are updated based on a fixed distribution are reflective of that distribution. Therefore, after the server transmits the initial weights throughout the network, each client pre-trains its model for some E epochs and broadcasts its updated weights to the server to be used for measuring the similarities. After the server pools all the updated weights, several distance metrics can be used to quantify this weight similarity. Here, cosine similarity is used that appears to deduce the optimal number of clusters more consistently compared to other metrics. The combination of hierarchical clustering using cosine similarity along with a threshold of t = 0.9, provided the best results in most cases; however, this highly depends on problem setting. Regarding hyperparameter, it is advisable to compare the framework's performance with respect to different values for t. As for the rest of the framework, we do not deviate from the standard FL protocol previously described. Following the cluster formation, each client trains its model instance for one epoch on its local data and transmits its updated weights to the server. Then, the server performs the personalized aggregation via model mixing, using only the weights of the clients that correspond to each cluster. Finally, the server transmits the client-specific models within the network, repeating this process until the framework converges. Figure 1 illustrates the described framework in the form of pseudocode.

/* Runs on Server-Side */
for each client $k \in \{1, 2, \dots K\}$ **do**
  Transmit h^0 // Initial weights
  Receive h_k^E // Pre-trained client weights
end for
Calculate clients' **cosine similarities** using h_k^E
Cosine Similarity **Hierarchical Clustering** to form C clusters
for each cluster $c \in \{1, 2, \dots C\}$ **do**
  $h_c^0 \leftarrow \sum_{k=1}^{|c|} \frac{|D_k|}{|D_c|} h_k^E$ // Aggregate cluster-level initial weights
  for each client $k \in c$ **do**
    Transmit h_c^0
  end for
end for
for each global round $t = 0, 1, \dots T-1$ **do**
  for each client $k \in \{1, 2, \dots K\}$ **do**
    Receive h_k^{t+1} // Client's updated weights
  end for
  for each cluster $c \in \{1, 2, \dots C\}$ **do**
    $h_c^{t+1} \leftarrow \sum_{k=1}^{|c|} \frac{|D_k|}{|D_c|} h_k^{t+1}$ // Aggregate cluster-level global mode
    for each client $k \in c$ **do**
      $h_k^{t+1} \leftarrow \lambda h_k^{t+1} + (1-\lambda) h_c^{t+1}$ // Aggregate client-specific mode
      Transmit h_k^{t+1}
    end for
  end for
end for

/* Runs on Client-Side */
$h_k^0 = h^0$ //Initialize local model with received weights
for each pretraining round $e = 0, 1, \dots E-1$ **do**
  $h_k^{e+1} = h_k^e - \eta_e \bar{\nabla} \ell(h_k^e(\mathbf{x}), y)$ // Update local model
end for
Transmit h_k^{e+1} // Pre-trained local weights
Receive h_c^0 // Cluster-level initial weights
$h_k^{e+1} = h_c^0$ // Load cluster-level initial weights
for each global round $t = 0, 1, \dots T-1$ **do**
  $h_k^{t+1} = h_k^t - \eta_t \bar{\nabla} \ell(h_k^t(\mathbf{x}), y)$ // Update local model
  Transmit h_k^{t+1} // The updated local weights
  Receive h_k^{t+1} // The aggregated personalized weights
end for

Figure 1: An overview of our framework in pseudocode.

4 Experiments

Throughout experimentations, the positive effect of combining CFL and PFL in real-world applications is shown, where no prior knowledge of clients' data and tasks is assumed. Task diversity and distribution heterogeneity is present within the federation, and maximizing personalization is the goal. Throughout

our experiments, the following issues are addressed: (1) the number of layers' weights for computing the similarities, (2) the pre-training time for the weights to adequately reflect the underlying distribution and (3) the advantages of combining PFL and CFL with respect to the framework's performance regarding personalization. For the first two tasks, 'similarity experiments' are designed that do not relate with the federated setting, while for the third issue the federated protocol is simulated.

4.1. Model Architecture and Dataset

Regarding model architecture, a CNN is utilized that is comprised of 5 ReLU activated convolutional layers, 2 max pooling layers and 1 fully connected layer, deriving in a total model size of 511 KBs. Regarding the datasets used throughout the experiments, we opted for CIFAR-100 comprised of 100 classes and CIFAR-10 comprised of 10 classes, where each class on both datasets has 600 32x32 RGB images. To study the non-IID setting, we must pathologically create one. We define a parameter $a_1 \in [0,1]$, where for instance $a_1 = 0.8$ means that 80% of the total data of a client are for one class and the rest $1 - a_1 = 20\%$ of data are reserved for other classes. Similarly, we also denote $a_2 \in [0,1]$, where for instance if $a_1 = 0.8$ and $a_2 = 0.6$, this means that 80% of a client's data is reserved for one class, $(1 - a_1) * a_2 = 12\%$ is reserved for the second class and $(1 - a_1) - ((1 - a_1) * a_2) = 8\%$ are reserved for other classes. As a result, we can pathologically create a non IID environment with imbalanced local distributions, that closely resembles a real-world scenario.

4.2. Experimental Setup

For the similarity experiments, CIFAR-10 with 10 clients and 3 tasks per client was used, with $a_1 = 0.6$ and $a_2 = 0.8$. For the federated setting simulation, 16 clients were employed as well as two pathological Non-IID settings, with imbalanced local distributions, using CIFAR-100 classes with $a_1 = 0.6$ and $a_2 = 0.8$, as well as with $a_1 = 0.7$ and $a_2 = 0.8$. Firstly, simulating a scenario where all tasks within a cluster are shared by all its clients, two tasks were assigned per client for a total of 16 tasks with the optimal number of clusters being 8. Conversely, a scenario was simulated in which, within each cluster there are tasks that do not concern all clients. Here, 3 tasks were assigned per client for a total of 20 tasks within the network, with 4 optimal clusters. The implementation was carried out using PyTorch for training and Python's socket interface for networking. Training lasted 500 global training rounds, using cross-entropy loss and stochastic gradient descent with a learning rate of 0.01.

5 Results on Federated Simulation Experiments

During the federated simulation experiments, the following issues were addressed: (1) the way to determine a good value for the mixing parameter λ and (2) an effectiveness comparison of clustered PFL in large-scale heterogeneous settings w.r.t. FL and PFL. Figure 2 illustrates the validation loss per training round, comparing both different mixing values (λ), as well as the effectiveness of clustering. Based on the two graphs, a striking difference in oscillations is observed regarding validation loss w.r.t. clustering. Our analysis outcome is that clustering significantly reduces these oscillations and smoothens the loss's convergence, by preventing the models from drifting too much from their local distributions. Additionally, clustered clients reach lower convergence points, indicating a better fit on the data and highlighting the effectiveness of the method in such settings. Regarding the mixing parameter λ, no specific value outshines regarding validation performance, with all of them converging to roughly the same point.

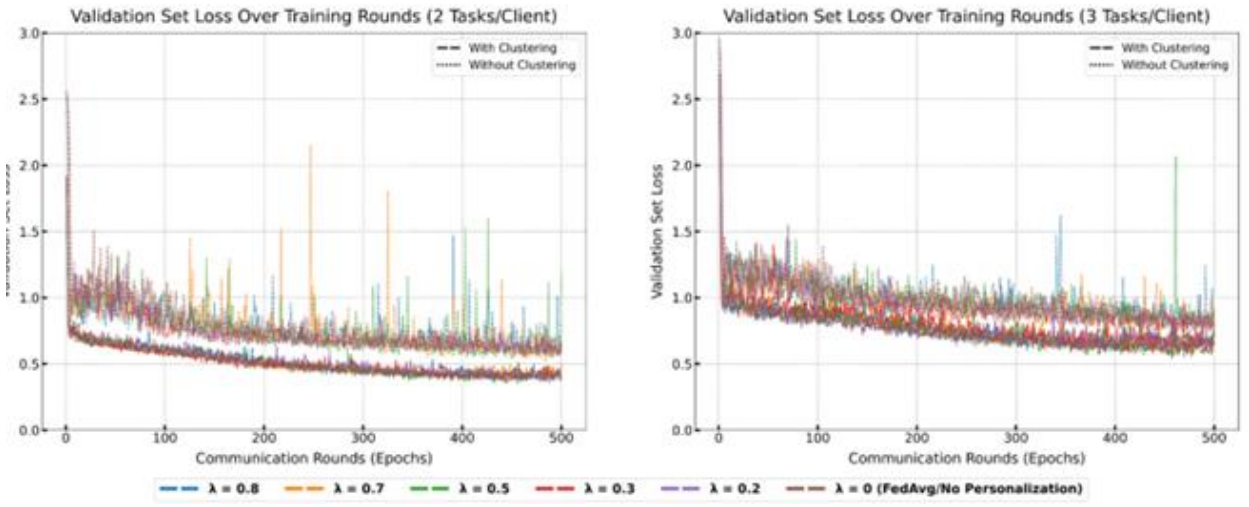

Figure 2. Validation loss per training round, comparing both different mixing values (λ), as well as the effectiveness of clustering.

Figure 3, on the other hand, illustrate the test accuracy per training round, comparing both different mixing values (λ), as well as the effectiveness of clustering. It is obvious that the clustering framework steadily outperforms its counterparts, reaching an average of +12% and +11% in the two graphs respectively, in terms of test set accuracy. Furthermore, the striking difference between FedAvg (no clustering λ=0 or brown dotted) compared to all other approaches, highlights the need for personalization in large-scale heterogeneous embedded settings and the reason that one global model is usually not sufficient. Regardless of clustering or not, no particular λ stands out compared to others, besides FedAvg, which does not personalize and is the worst performing. Interestingly though, on the 2-tasks experiment, the highest recorded precision belongs to FedAvg (see Figure 4). This suggests, that in Non-IID and imbalanced settings, constructing only one global model will also result in a bias towards the majority tasks, leading to fewer predictions for underrepresented tasks, which tend to be correct and hence inflate the model's precision. Lastly, a general rule for choosing an appropriate λ value is the following. As λ increases we incorporate more of the locally trained model in each aggregation and vice versa. Thus, the more we expect a local distribution to diverge from its cluster level or global distribution, the greater λ should be.

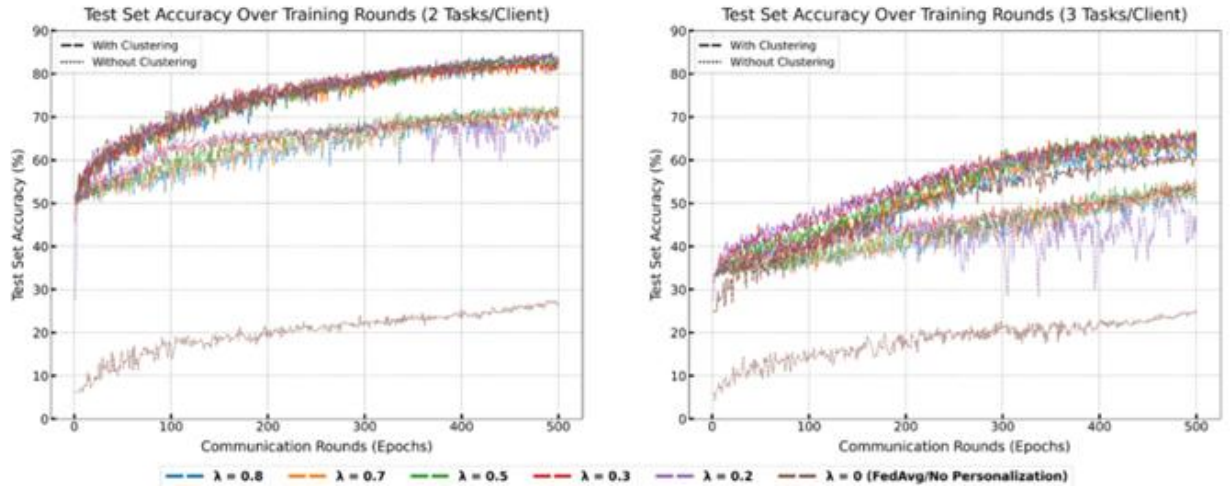

Figure 3. Test accuracy per training round, comparing both different mixing values (λ), as well as the effectiveness of clustering

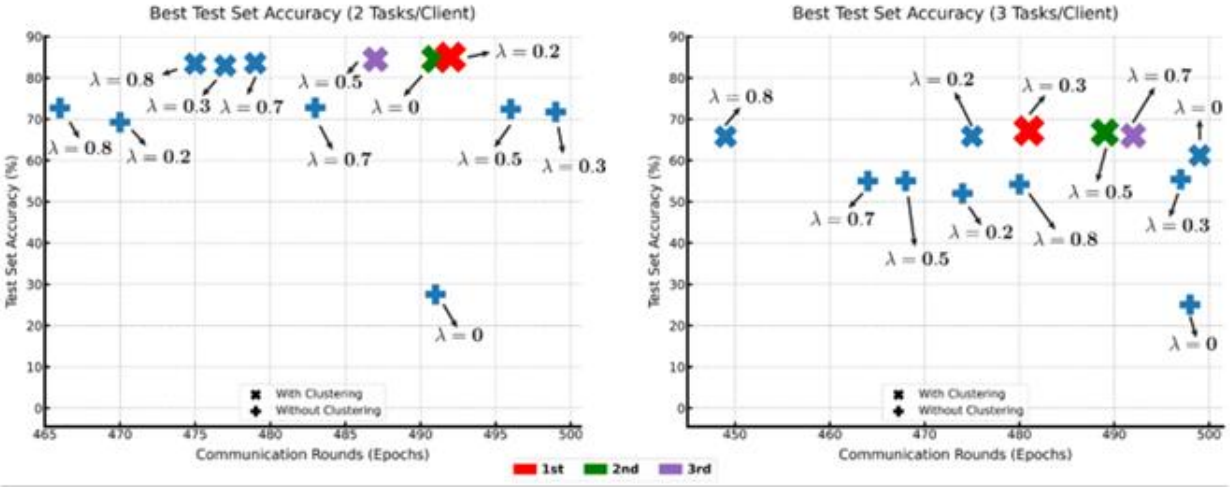

Figure 4. Top test set accuracy for each lambda, clustering or not, along with the training round it occurred.

Acknowledgments

This research has been co-financed by the Digital Transformation of Small and Medium Enterprises Initiative of the European Union (NextGeneration EU) and Greek national funds (Ellada 2.0) through the program "Development of Digital Product and Services", project acronym "Product Watch" (project code: 104702).

References

Chelsea Finn, P. A. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. Retrieved from https://arxiv.org/abs/1703.03400

Farzin Haddadpour, M. M. (2019). On the Convergence of Local Descent Methods in Federated Learning. Retrieved from https://arxiv.org/abs/1910.14425

Felix Sattler, K.-R. M. (2019). Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints. Retrieved from https://arxiv.org/abs/1910.01991

Filip Hanzely, P. R. (2020). Federated learning of a mixture of global and local models. Retrieved from https://arxiv.org/abs/2002.05516.

H. Brendan McMahan, E. M. (2016). Communication-Efficient Learning of Deep Networks from Decentralized Data. Retrieved from https://arxiv.org/abs/1602.05629

Virginia Smith, C.-K. C. (2018). Federated multi-task learning. Retrieved from https://arxiv.org/abs/1705.10467

Yishay Mansour, M. M. (2020). Three Approaches for Personalization with Applications to Federated Learning. Retrieved from https://arxiv.org/abs/2002.10619.