ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

FedHP: Federated Learning with Hyperspherical Prototypical Regularization

Samuele Fonio[1], Mirko Polato[1], Roberto Esposito[1] *

1- University of Turin - Computer Science Department
Corso Svizzera 175, Turin, Italy.

Abstract.
This paper presents FedHP, an algorithm that amalgamates federated learning, hyperspherical geometries, and prototype learning. Federated Learning (FL) has garnered attention as a privacy-preserving method for constructing robust models across distributed datasets. Traditionally, FL involves exchanging model parameters to uphold data privacy; however, in scenarios with costly data communication, exchanging large neural network models becomes impractical. In such instances, prototype learning provides a feasible solution by necessitating the exchange of a few class prototypes instead of entire deep learning models. Motivated by these considerations, our approach leverages recent advancements in prototype learning, particularly the benefits offered by non-Euclidean geometries. Alongside introducing FedHP, we provide empirical evidence demonstrating its comparable performance to other state-of-the-art approaches while significantly reducing communication costs.

1 Introduction

Federated Learning (FL) [1] is a revolutionary paradigm in distributed machine learning, founded on the principle of sharing models rather than raw data. In this framework, multiple parties with aligned objectives aim to leverage machine learning techniques to address specific tasks. The approach is motivated by the fact that machine learning approaches in general and deep learning models in particular often demand vast amounts of data to achieve robust generalization and good performance. Collaborating with other parties enables access to larger datasets, potentially enhancing outcomes significantly. However, concerns regarding data privacy hinder direct data sharing.

FL solves this dilemma by facilitating collaboration among parties while safeguarding data privacy. In its most basic form [1], each participant, or "client", in a federation trains a model locally. Subsequently, they share their trained model with a trusted central server. This server aggregates the received models into a global model and disseminates it back to the clients. This iterative process continues until convergence of the models is achieved.

Personalized Federated Learning (PFL) [2] emerges as one of the most practical and promising scenarios within FL. In PFL, clients prioritize improving the

*This work has been partly supported by the Spoke "FutureHPC & BigData" of the ICSC - Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union - NextGenerationEU.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

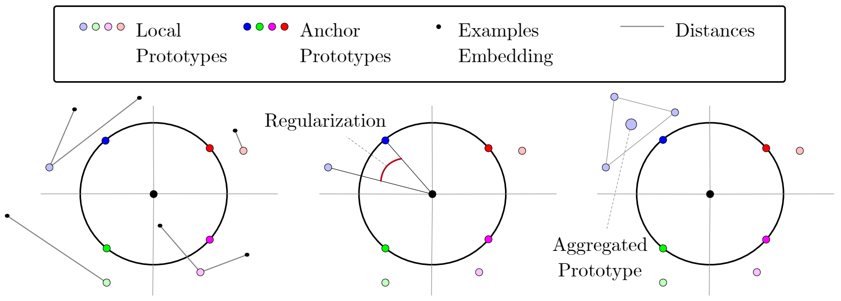

Fig. 1: The figure shows the main components of FedHP. On the left we emphasize the importance of computing distances from the prototypes which allow the inference of class probabilities. In the middle, we show that the regularization term is based on the cosine distance between anchor and local prototypes. On the right image we represent the aggregation of the prototypes on the server.

performance of their local models over attaining a global model with broad generalization capabilities. Notably, clients within the federation may exhibit highly diverse data distributions, reflecting the real-world heterogeneity encountered in many applications. A recent promising PFL approach [3] is based on Prototype Learning (PL), which showed to achieve comparable results to state-of-the-art approaches while requiring a fraction of communication resources.

In modern deep learning based prototype learning approaches the prototypes are built in an embedding space learnt by a deep learning model [4]. One main challenge when this approach is used in FL systems is about the *alignment of the features*. In fact, local distributions might bring to very different representations in the embedding space, making the aggregation of the models unreliable. To overcome this problem, some methods have proposed regularization terms in order to obtain alignment among the representations. For example FedProto [5] leverages a regularization term that seeks to align local and global prototypes. Dai et al. [6] employ a specific aggregation rule in order to inject personalized contributions in the global model. However, this method shares both the model and the prototypes, thus losing some of the attractive properties of prototype federated learning systems.

Here, we introduce FedHP, an approach that learns local prototypes and avoids the local drift leveraging ideas firstly introduced by Hyperspherical Prototypical networks [7]. Our contribution with this work is threefold: (*i*) we propose an efficient method to learn the prototypes positions alongside the training of the embedding model, which allows the embedding and the prototypes to be mutually optimized to work well together; (*ii*) we propose a regularization term based on hyperspherical prototypical networks [7] that aims at dividing the space in a uniform way, and (*iii*) we provide empirical evidence showing that FedHP is not demanding in terms of communication costs, it is competitive with non-prototype based approaches and easily outperforms state-of-the-art prototype techniques.

2 Method

Let us assume to be given a dataset $X = \{(\mathbf{x}_i, y_i)\}$, with $\mathbf{x}_i$ taking values in a sample space $\mathcal{X}$, $y_i \in \mathcal{C} = \{1 \dots C\}$, and $|X| = N$. To describe *FedHP*, let us start by noticing that in PL, the prototypes are usually defined as the average of the representations belonging to a certain class [3]. This choice is very common also in federated learning approaches using prototype learning [5, 6], with the twist that the computation of prototypes is weighted by the number of examples for each class present in each client. In our approach, following [4, 8], we define the prototypes as parameters of the neural network. This choice is computationally efficient and avoids computing the per-class average of all examples at every batch. Specifically, a backbone network $f(\cdot, \theta) : \mathcal{X} \to \mathbb{R}^d$ is augmented with parameters $\Pi = \{\boldsymbol{\pi}_j : j \in \mathcal{C}\}$ representing the prototypes, with $\boldsymbol{\pi}_j \in \mathbb{R}^d$. The local objective that each client minimizes is:

$$\arg\min_{\theta, \Pi} \mathcal{L}(\theta, \Pi; \lambda),$$

i.e., each client searches for the set of parameters θ and Π that minimize the regularized loss $\mathcal{L}$ over the training set X:

$$\mathcal{L}(\theta, \Pi; \lambda) = \mathcal{L}_D(\theta, \Pi) + \lambda \mathcal{L}_H(\Pi), \tag{1}$$

where $\lambda \geq 0$ is an hyperparameter of the method. The first term $\mathcal{L}_D$ is a distance based cross-entropy loss [4]:

$$\mathcal{L}_D(\theta, \Pi) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in X} - \log \frac{e^{-d(\mathbf{z}_i, \pi_{y_i})}}{\sum_{j \in \mathcal{C}} e^{-d(\mathbf{z}_i, \pi_j)}}, \tag{2}$$

where $\mathbf{z}_i = f(\mathbf{x}_i; \theta) \in \mathbb{R}^d$, f is the output of the backbone network and $d(\cdot, \cdot)$ is the euclidean distance (see also Figure 1, left diagram).

In FL scenarios, the main issue with prototype learning is the tendency for prototypes to disalign, as each client learns their own prototype embedding that can diverge over time. This becomes particularly difficult in strongly heterogeneous scenarios, where the local distribution differs consistently. To overcome this problem, our method leverages techniques to spread prototypes uniformly [7]. In particular, we initialize the prototypes to be uniform on the hypersphere $S = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|^2 = 1\}$ by minimizing the *Uniform* loss $\mathcal{L}_U$ [7]. Specifically, we search for prototype parameters Π that minimize:

$$\mathcal{L}_U(\Pi) = \frac{1}{C} \sum_{i \in \mathcal{C}} \max_{j \in \mathcal{C}} M_{ij}, \quad \mathbf{M} = \mathbf{P}\mathbf{P}^\top - 2\mathbf{I}, \quad \text{s.t. } \forall i \, \|\mathbf{P}_i\| = 1, \tag{3}$$

where $\mathbf{P}_{i,\cdot} = \boldsymbol{\pi_i}$ is a matrix containing the current set of hyperspherical prototypes, $\mathbf{I}$ denotes the identity matrix and $\mathbf{M}$ contains the pairwise cosine similarities between prototypes. To summarize: the prototypes are initialized on the server by minimizing (3) and then broadcasted to the clients. When the

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

initialization phase ends, the initial prototypes are fixed as anchor points $\bar{\boldsymbol{\pi}}$ in the embedding space, and they are then used to avoid local driftings of the prototypes by means of the $\mathcal{L}_H$ term in the main loss function $\mathcal{L}$:

$$\mathcal{L}_H(\Pi) = \sum_{j \in \mathcal{C}} 1 - \frac{\bar{\boldsymbol{\pi}}^j \cdot \boldsymbol{\pi}^j}{\|\bar{\boldsymbol{\pi}}^j\| \|\boldsymbol{\pi}^j\|} \tag{4}$$

i.e., the sum, for all classes j, of the cosine distance between the anchor prototypes $\bar{\boldsymbol{\pi}}^j$ and local prototypes $\boldsymbol{\pi}^j$ (see Figure 1, middle diagram). It is worth noting that, thanks to this loss function, we encourage the prototypes to keep a uniform partition of the space.

In FedHP, at each local training step, the prototypes are updated to minimize (1) and are then sent to the server. Upon receiving the prototypes, the server aggregates them, weighting their contribution according to the number of examples of each class that each client owns. The result is a new global set of prototypes $\{\boldsymbol{\pi}_g^j\}_{j=1}^C$, with:

$$\boldsymbol{\pi}_g^j = \frac{1}{K_t} \sum_{i=1}^{K_t} \frac{N_i^j}{N_i} \boldsymbol{\pi}_i^j \tag{5}$$

where, K_t is the number of clients selected at round t, N_i^j is the subset of examples in client i labelled as j, N_i is total number of examples in client i, and $\boldsymbol{\pi}_i^j$ is the j-th local prototype in client i (see Figure 1, right diagram). The global set of prototypes are then sent to each client and each client substitutes its local prototypes with the received ones. This results in a very low communication overhead, since the only parameters to be shared are the prototypes and not the model. Learning the prototypes through our framework, we obtain a smooth alignment of the prototypes and a fine partition of the space, avoiding prototypes drifting.

3 Experiments and Discussion

We compare FedHP with three other methods. We selected FedAvg to provide a baseline comparison with methods that share the full model at each iteration. FedProto [5] is the current state-of-the-art method among federated learning methods that shares only the prototypes. FedNH [6] is an hybrid method that shares both the prototypes and the model at each iteration.

We simulate non-iid scenarios in two ways: with a label skew governed by a Dirichlet distribution [9] with parameter $\beta \in \{0.3, 1.0\}$. We test the four algorithms on MNIST, Cifar10 and SVHN. We do not perform preprocessing on the datasets with the exception of feature normalization. In the case of MNIST, we tested a simple convolutional network [1], for Cifar10 and SVHN we employed the network from [6]. We experiment with 100 clients and a participation rate of 10%. All the algorithms ran for 150 rounds in the case of MNIST and Cifar10. On SVHN, we ran the algorithms for 300 rounds. At each round, we let the

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Table 1: Percentage of test accuracy on 3 datasets for our proposed method and the competing approaches. The results displayed are the mean over 5 runs with their standard deviation. We put in bold the best results between the approaches sharing only the prototypes and underlined the overall best.

	MNIST		
	IID	Dir(0.3)	Dir(1.0)
FedAvg	$\underline{97.75}$ ± 0.25	$\underline{97.61}$ ± 0.21	$\underline{97.99}$ ± 0.07
FedNH	84.30 ± 21.10	96.42 ± 3.45	90.16 ± 11.08
FedProto	84.40 ± 0.67	85.58 ± 1.25	85.52 ± 0.68
FedHP	**96.04** ± 0.14	**96.82** ± 0.21	**96.53** ± 0.10
	Cifar10		
FedAvg	$\underline{60.27}$ ± 0.74	56.11 ± 1.84	$\underline{60.43}$ ± 1.05
FedNH	52.17 ± 1.61	$\underline{65.65}$ ± 1.32	$\underline{58.39}$ ± 1.32
FedProto	35.42 ± 0.47	49.15 ± 0.94	41.16 ± 0.98
FedHP	**45.57** ± 0.65	**62.19** ± 0.86	**53.00** ± 0.98
	SVHN		
FedAvg	$\underline{90.27}$ ± 0.10	88.33 ± 0.72	$\underline{89.95}$ ± 0.95
FedNH	86.33 ± 0.86	$\underline{89.89}$ ± 1.25	86.32 ± 0.43
FedProto	59.72 ± 0.64	64.70 ± 1.81	61.53 ± 0.88
FedHP	**82.11** ± 0.30	**73.50** ± 2.72	**80.64** ± 1.36

local minimizers to run for 5 local epochs, using SGD, a learning rate of 0.01, a momentum of 0.9, and weight decay set to 10^{-4} for parameters θ and Adam with learning rate 0.005 for the prototypes Π. Each client keeps 20% of their dataset as local test set. During the last round, the global model is broadcast to all the clients, which perform one last fit on their local data. This is standard practice in a PFL scenario. The parameter λ in (1) is set to 0.1. The latent embedding dimension for our method and FedProto is 1024 for MNIST and 1600 for Cifar10 and SVHN. It is worth mentioning that FedNH and FedHP only use the backbone network, while FedAvg and FedProto rely on the complete networks (backbone + head).

Based only on the quantity of information shared at each iteration, we would assume that FedNH and FedAvg be dominant in our experiments and this is indeed the case in many situations. However, as shown in Table 1, FedHP largely outperforms FedProto and is often only slightly worse (when not better) than the other two methods. In addition, the communication overhead makes methods like FedAvg and FedNH impractical in scenarios where communication needs to be limited to the bare minimum such as in some edge devices and IoT/embedded systems. Sharing only the prototypes effectively overcomes this challenge. Notably, at each round, FedAvg clients share 582,026 parameters for MNIST and 725,952 for Cifar10 and SVHN, whereas FedNH incurs an even higher cost due to exchanging both the model and the prototypes. On the contrary, our approach, as well as FedProto, drastically reduces communication costs to 10,240 parameters for MNIST and 16,000 for Cifar10 and SVHN.

Leveraging the capabilities of Hyperspherical Prototypical Networks, we capitalize on the continual refinement of prototypes. Specifically, this strategy enables the integration of hyperspherical regularization and batch-level prototype updates, resulting in higher performance and faster training. These outcomes also highlights the pivotal role of a uniform spatial separation through hyperspherical regularization.

4 Conclusions

In this paper, we have introduced FedHP a new method that leverages the effectiveness of Prototype Learning in a Federated Learning scenario. FedHP employs recent techniques from the Prototype Learning research, defining the prototypes as parameters of the network and using a regularization term inspired by Hyperspherical Prototypical Networks. FedHP showed promising results outperforming the state-of-the-art competitor which shares only prototypes, and remaining competitive with approaches sharing the whole model.

In the future, we will investigate how to improve performances of FedHP by augmenting the quantity of information shared, while still avoiding to share the complete model and without compromising the privacy of the technique. A possible direction could be extending FedHP following the sharing scheme of LG-FedAvg[10] which only shares the head of the network.

References

[1] Brendan McMahan et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Alysa Ziying Tan et al. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[3] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NeurIPS*, 30, 2017.

[4] Hong-Ming Yang et al. Robust classification with convolutional prototype learning. In *CVPR*, pages 3474–3482, 2018.

[5] Yue Tan et al. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI*, volume 36, pages 8432–8440, 2022.

[6] Yutong Dai et al. Tackling data heterogeneity in federated learning with class prototypes. In *AAAI*, volume 37, pages 7314–7322, 2023.

[7] Pascal Mettes, Elise Van der Pol, and Cees Snoek. Hyperspherical prototype networks. *NeurIPS*, 32, 2019.

[8] Loic Landrieu and Vivien Sainte Fare Garnot. Leveraging class hierarchies with metric-guided prototype learning. In *BMVC*, 2021.

[9] Qinbin Li et al. Federated learning on non-iid data silos: An experimental study. In *arXiv 2102.02079*, 2021.

[10] Paul Pu Liang et al. Think locally, act globally: Federated learning with local and global representations. In *NeurIPS Workshop on Federated Learning*, 2019.