Prototype Guided Federated Learning of Visual Feature Representations

Umberto Michieli[1,2*] Mete Ozay[1]
[1]Samsung Research UK [2]University of Padova
{u.michieli, m.ozay}@samsung.com

Abstract

Federated Learning (FL) is a framework which enables distributed model training using a large corpus of decentralized training data. Existing methods aggregate models disregarding their internal representations, which are crucial for training models in vision tasks. System and statistical heterogeneity (e.g., highly imbalanced and non-i.i.d. data) further harm model training. To this end, we introduce a method, called FedProto, which computes client deviations using margins of prototypical representations learned on distributed data, and applies them to drive federated optimization via an attention mechanism. In addition, we propose three methods to analyse statistical properties of feature representations learned in FL, in order to elucidate the relationship between accuracy, margins and feature discrepancy of FL models. In experimental analyses, FedProto demonstrates state-of-the-art accuracy and convergence rate across image classification and semantic segmentation benchmarks by enabling maximum margin training of FL models. Moreover, FedProto reduces uncertainty of predictions of FL models compared to the baseline. To our knowledge, this is the first work evaluating FL models in dense prediction tasks, such as semantic segmentation.

1. Introduction

Federated Learning (FL) is a framework proposed for distributing training of machine learning models in a network of clients (devices) with local data processed only at clients [4, 24, 30, 59]. In FL, models are trained across multiple rounds. At the beginning of each round, every participating client receives an initial model from a central server, optimizes the model on its local training data and sends the updated model back to the server. The server then aggregates all the models and updates the aggregate model [35].

Challenges of FL: Training models in FL systems introduces several novel challenges [24, 30]. In this work, we address problems caused by system and statistical heterogeneity. System heterogeneity refers to variable computational (*e.g.*, CPU, memory, battery level) and communication (*e.g.*, wifi) capabilities of each device [41, 54]. Early approaches suggest to drop devices that fail to compute predetermined workloads within a time window [4, 35]. However, Li *et al*. [31] showed that this has negative effects on convergence as it limits the number of effective devices contributing to training and may induce bias, if dropped devices have specific data characteristics. Hence, we tolerate partial workload on clients following recent works [31, 42].

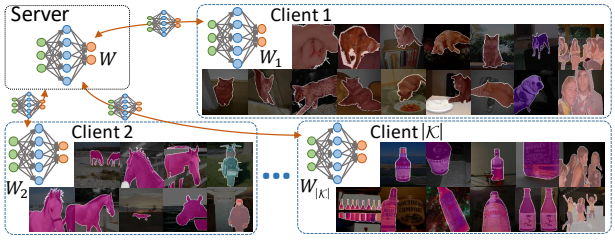

Figure 1. Visual data observed at distributed clients $k \in \mathcal{K}$ are non-i.i.d. and imbalanced. This represents a challenge for federated learning of vision models with parameters $W_k, \forall k$.

Statistical heterogeneity reflects another major challenge for convergence: whilst in centralized training, data can be assumed independent and identically distributed (i.i.d.), decentralized data is generally highly imbalanced (*e.g.*, local data may contain different number of samples for different classes on each device) and non-i.i.d. (*e.g.*, samples in remote clients may have large correlation due to user-specific habits or preferences) [63], as depicted in Fig. 1.

Challenges of FL of visual feature representations: Representation learning has been a prosperous technique used to perform complex computer vision tasks, such as image classification and segmentation [3, 14]. In this paradigm, a model is trained to learn *rich* feature representations of its inputs, and learned representations are employed by task specific predictors (*e.g.*, classifiers or detectors). Current FL approaches focus on learning features by considering only statistical properties of data, such as joint distribution of samples and their class labels [20], and weights of models [31]. In FedAvg [35], weights are aggregated with importance scores proportional to size of lo-

*Researched during internship at Samsung Research UK.

cal datasets, ignoring the learning dynamics. A similar approach has been followed by many subsequent methods [16, 20, 25, 31]. More recently, increasing interest has been devoted toward elucidating aggregation procedures. Attention methods [21, 22, 40, 57, 60] were proposed using functions of difference between parameters of local and aggregate models. However, these works disregard relationship between statistical properties of the learned representations.

Here, instead, we propose a prototype guided federated optimization method (FedProto), which leverages the model aggregation procedure by prioritizing distributed models on basis of their learned prototypical representations of object categories. In particular, FedProto consists of three steps:

(i) Prototypical representations: First, we compute prototypical representations using local and aggregate models motivated by their success in meta-learning [2, 6, 26, 29, 33, 48], domain adaptation [39, 49, 50], semantic segmentation [11, 55] and continual learning [36, 62].

(ii) Confidence of local and aggregate models: Second, we compute confidence of local and aggregate models with respect to their decision on local data using prototypical (hypothesis) margins (PMs). PMs have been explored for developing learning vector quantization (LVQ) methods [17, 23, 37, 45, 46]. In [10], PMs are shown to lower bound sample margins and provide a rigorous upper bound of generalization error. Unlike PMs proposed for individual models, we aim to measure the change in semantic representations of FL models learned at different clients and over different rounds considering their generalization properties. Therefore, we first define a novel semantic PM. Then, motivated by these theoretical results, we drive the model aggregation process combining a confidence measure computed between local prototypes at the beginning and at the end of the local optimization (*i.e.*, *Local PM*), as well as a measure computed between aggregate and local prototypes at the server-level (*i.e.*, *Aggregate PM*). Although margins between features and prototypes have been used to solve other vision tasks (*e.g.*, few-shot learning [29]), to our knowledge, our work is the first to compute margins among sets of prototypes and employ them for federated optimization.

(iii) Prototype-based weight attention: Finally, we propose a weight attention mechanism during global aggregation of local models using non-linear functions (*e.g.*, sigmoid) of prototypical margins. In FL, state-of-the-art attention methods [22, 57] consider only statistics of local models ignoring their effect on decision boundary. Instead, our attention mechanism quantifies this information by margins and employs it for aggregation. We conjecture that our mechanism enables maximization of latent-level margins in FL, which is experimentally justified in Sec. 5.

Intuitively, driving the model to focus on class prototypes, we achieve a better shaping of the inner space (thus acting as regularization constraint) that eventually eases the classifier task, which is a harder task than feature extraction [56]. Therefore, FedProto shows better convergence rate and accuracy, ultimately achieving a closer latent space organization to the one centralized training would produce.

The main contributions of this paper are as follows:

- We propose a novel FL algorithm (FedProto) which applies a *prototypical margin-based model attention mechanism* to drive FL optimization in heterogeneous systems.
- We achieve state-of-the-art results on a variety of image classification and semantic segmentation benchmarks. To the best of our knowledge, this is the first work exploring federated learning of semantic segmentation models.
- We propose two quantitative metrics and a qualitative method based on entropy maps to analyse statistical properties of feature representations learned in FL systems.

2. Federated Learning

In an FL system consisting of a set of clients $\mathcal{K} = \{1, 2, \dots, K\}$, parameters $W_k \in \mathcal{W}_k$ of models $M_k : \mathcal{W}_k \times \mathcal{X}_k \rightarrow \mathcal{Y}_k$, are optimized at each client $k \in \mathcal{K}$ using its local dataset to learn feature representations, where $\mathcal{X}_k = \{\mathbf{x}_{k,j}\}_{j=1}^{n_k}$ and $\mathcal{Y}_k = \{\mathbf{y}_{k,j}\}_{j=1}^{n_k}$ denote respectively the set of samples and their ground truth labels (*e.g.*, one-hot encoded vectors of category labels for image classification, and vectors of segmentation maps for image segmentation) observed at the client k. In centralized FL systems, a central server coordinates the optimization of a set of parameters $\mathcal{W}$ of an aggregated model $M(\mathcal{W}, \cdot)$ by minimizing a global learning objective $L(W)$ [35] without sharing local datasets $\mathcal{S}_k = \{s_{k,j} = (\mathbf{x}_{k,j}, \mathbf{y}_{k,j})\}_{j=1}^{n_k}$ by solving

$$\min_{W \in \mathcal{W}} L(W) = \min_{W \in \mathcal{W}} \sum_{k \in \mathcal{K}} p_k L_k(W; \mathcal{S}_k), \tag{1}$$

where the local objective is computed by

$$L_k(W; \mathcal{S}_k) = \frac{1}{n_k} \sum_{j=1}^{n_k} l_k(W; s_{k,j} \in \mathcal{S}_k), \tag{2}$$

with $l_k(\cdot;\cdot)$ being a user-specific loss function, $p_k \geq 0$ is the weight of the objective $L_k(\cdot;\cdot)$ of the k^{th} client and $\sum_{k \in \mathcal{K}} p_k = 1$. McMahan *et al.* [35] proposed to use $p_k = \frac{n_k}{n}$, where $n = \sum_{k \in \mathcal{K}} n_k$. Thereby, $L(W)$ coincides with the training objective of the centralized setting.

Federated averaging (FedAvg) [32, 35] is a benchmark federated optimization algorithm widely used to solve the problem (1). In FedAvg, a subset $\mathcal{K}^t \subseteq \mathcal{K}$ of K' clients are selected according to p_k at each federated round t. Selected clients $k \in \mathcal{K}^t$ download the aggregated model $W^t \in \mathcal{W}^t$ from a central server, perform local optimization minimizing an empirical objective $L_k(W^t; \mathcal{S}_k)$ with learning rate η for F epochs using a local optimizer such as SGD, and then send the final solution W_k^{t+1} back to the server. The server averages the solutions obtained from the clients with weights proportional to the size of the local datasets by

$$W^{t+1} = \sum_{k \in \mathcal{K}^t} \frac{n_k}{\sum_{j \in \mathcal{K}^t} n_j} W_k^{t+1}. \tag{3}$$

The procedure is iterated for $T-1$ federated rounds and the final aggregate model is then identified by W^T.

Optimizing local models with the same number of local epochs at all clients is unfeasible for real-world applications [4, 20, 31]. A more natural approach is to allow the epochs to vary according to the characteristics of the FL system, and to properly merge solutions accounting for heterogeneity of the system, as we formalize next.

3. Prototype Guided Federated Learning

Prototypical representations have been successfully employed in various computer vision tasks [26, 36, 39, 48]. In this work, we employ prototypes for federated optimization of vision models. Our prototype guided federated optimizer (FedProto) is motivated by the results obtained from the recent theoretical and experimental analyses of generalization capacity of latent class-conditional prototypes [26, 48].

Partial workload toleration: First of all, we observe that different clients in FL systems are likely to have very different resource constraints (causing system heterogeneity), such as different data resources, hardware configurations (*e.g.*, visual sensors, cameras or processors), network connections and battery levels [24]. Therefore, we allow partial amount of work to be conducted locally by each client prior to the aggregation stage, as utilized in FedProx [31]. In other words, at each round, instead of dropping $\delta\%$ of clients that performed less epochs than the total number F in a predetermined amount of time, we aggregate all the solutions sent from local clients tolerating partial workload, *i.e.*, even if the completed number of local epochs is $F' < F$. Following [31], we mimic this behavior by uniformly sampling $F' \sim \mathcal{U}([0, F))$ on each client.

At each round t and client k, a local model $M_k^t(\mathcal{W}_k^t; \mathcal{X}_k) = C_k^t(\mathcal{W}_{c,k}^t) \circ E_k^t(\mathcal{W}_{e,k}^t; \mathcal{X}_k)$ is computed, where $\circ$ denotes function composition, and $\mathcal{W}_{c,k}^t \subset \mathcal{W}_k^t$ and $\mathcal{W}_{e,k}^t \subset \mathcal{W}_k^t$ denotes sets of parameters of classifiers and encoders embodied in the model M_k^t, respectively. For each input $\mathbf{x}_{k,j} \in \mathcal{X}_k$, its latent representation $\mathbf{e}_{k,j}^t = E_k^t(\mathcal{W}_{e,k}^t; \mathbf{x}_{k,j})$ is computed and then fed to a classifier $C_k^t(\mathcal{W}_{c,k}^t; \mathbf{e}_{k,j}^t)$ to retrieve class-wise probability scores. Features corresponding to the same class are then averaged to construct local latent class-conditional prototypes.

3.1. Computation of Prototypes

At each round $t > 0$, class $c \in \mathcal{C}$ and client $k \in \mathcal{K}^t$, the c^{th} element of prototypes $\mathbf{p}_k^t$ is computed by

$$\mathbf{p}_k^t[c] = \sum_{\mathbf{e}_{k,j,c}^t \in \mathcal{F}_{k,c}^t} \frac{\mathbf{e}_{k,j,c}^t}{\mathbf{n}_k^t[c]}, \quad \forall c \in \mathcal{C}, \forall k \in \mathcal{K}^t, \tag{4}$$

where $\mathcal{F}_{k,c}^t$ is the set of feature vectors $\mathbf{e}_{k,j,c}^t$ extracted from the sample $\mathbf{x}_{k,j} \in \mathcal{X}_k$ belonging to the class c, and $\mathbf{n}_k^t[c] = |\mathcal{F}_{k,c}^t|$ is the cardinality of $\mathcal{F}_{k,c}^t$. At $t = 0$, we initialize prototypes as $\mathbf{p}_k^0[c] = \mathbf{0}$, $\forall c, k$. Since features representing different classes have variable norm [58], we employ min-max normalization over the channels and denote the normalized prototypes by $\hat{\mathbf{p}}_k^t$.

3.2. Local and Aggregate Prototype Margins

To guide the optimization, we rely on a combination of two clues derived from displacement of prototypes:

1. *Local Prototype Margin (LPM)* measures deviation of on-client prototypes before and after local training.
2. *Aggregate Prototype Margin (APM)* measures deviation of aggregate prototypes from local prototypes.

As a measure for displacement, we embraced the margin theory [10, 17, 29, 37, 43, 45], in which PMs measure the distance between features and class decision boundaries. In our work, instead, we aim to measure change of semantic representations among clients over different rounds for FL. Therefore, we propose a novel semantic PM next.

Definition 3.1 (Semantic PM - SPM)**.** Given two prototype vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ defined on the same class space $\mathcal{C}$, we restrict to $\mathcal{C}' \subset \mathcal{C}$ such that $\mathbf{n}_i[c] > 0$ and $\mathbf{n}_j[c] > 0$, $\forall c \in \mathcal{C}'$, the distance between prototypes corresponding to the same semantic label c is computed by

$$\mathbf{d}_{i,j}^+[c] = d(\mathbf{p}_i[c], \mathbf{p}_j[c]), \quad \forall c \in \mathcal{C}', \tag{5}$$

and the average distance between prototype of a certain class c and prototypes of different classes is computed by

$$\mathbf{d}_{i,j}^-[c] = \sum_{\substack{c' \neq c \\ c' \in \mathcal{C}'}} \frac{d(\mathbf{p}_i[c], \mathbf{p}_j[c'])}{|c' \neq c \wedge c' \in \mathcal{C}'|}, \quad \forall c \in \mathcal{C}'. \tag{6}$$

Then, the SPM for class c is defined by

$$\mu(\mathbf{p}_i[c], \mathbf{p}_j) \stackrel{def}{=} \frac{\mathbf{d}_{i,j}^-[c] - \mathbf{d}_{i,j}^+[c]}{\mathbf{d}_{i,j}^-[c] + \mathbf{d}_{i,j}^+[c]}, \quad \forall c \in \mathcal{C}'. \tag{7}$$

In FL, we employ SPMs in two cases, LPM and APM, which are defined in Def. 3.2 and 3.3. In the analyses, we identify $d(\cdot, \cdot)$ by the Euclidean distance, since it has been shown to outperform cosine similarity [11, 48] or to achieve comparable performance [38, 55].

Definition 3.2 (LPM)**.** The LPM is defined by

$$\boldsymbol{\mu}_{\text{loc},k}^t[c] \stackrel{def}{=} \mu(\hat{\mathbf{p}}_k^{t-1}[c], \hat{\mathbf{p}}_k^t), \quad \forall k \in \mathcal{K}^t, \forall c \in \mathcal{C} \tag{8}$$

and it measures change of local prototypes obtained from local models before and after their local training.

Definition 3.3 (APM)**.** The APM is defined to measure discrepancy between local and aggregate set of prototypes by

$$\boldsymbol{\mu}_{\text{agg},k}^t[c] \stackrel{def}{=} \mu(\hat{\mathbf{p}}_k^t[c], \hat{\mathbf{p}}_{\text{agg}}^{t-1}), \quad \forall k \in \mathcal{K}^t, \forall c \in \mathcal{C} \tag{9}$$

where the aggregate set of prototypes is defined by

$$\hat{\mathbf{p}}_{\text{agg}}^t[c] \stackrel{def}{=} \sum_{k \in \mathcal{K}^t} \frac{\mathbf{n}_k^t[c]}{\mathbf{n}_{\text{agg}}^t[c]} \hat{\mathbf{p}}_k^t[c], \quad \forall c \in \mathcal{C}, \tag{10}$$

with aggregate number of features $\mathbf{n}_{\text{agg}}^t[c] = \sum_{k \in \mathcal{K}^t} \mathbf{n}_k^t[c]$, $\forall c \in \mathcal{C}$ and initialization $\hat{\mathbf{p}}_{agg}^0[c] = \mathbf{0}$.

We remark that APM requires transmission of prototypes from clients to server. However, this does not raise privacy issues since prototypes represent only an averaged statistic over all local data of already compressed feature representations, nor large communication overhead, as the size of prototypes is negligible compared to the model size. While local deviation measured by LPM gives a hint of how much a model adapts its inner representation for each class, server-side deviation measured by APM tells how much a local model changes its inner representations with respect to the prototypical representations aggregated over previous rounds and clients. The effect of distributed versus centralized calculation of margins is analysed in Sec. 5.1.

3.3. Federated Attention using Prototype Margins

Client deviations are computed by summing over all the classes and applying a sigmoid function σ [37, 45] by

$$\mathbf{v}_{\iota}^{t}[k] = \sigma\left(\sum_{c\in\mathcal{C}} \boldsymbol{\mu}_{\iota,k}^{t}[c]\right), \quad \forall k \in \mathcal{K}^t, \iota \in \{\text{loc, agg}\}. \quad (11)$$

Definition 3.4 (Local, aggregate and federated attention)**.** A local (aggregate) weight attention vector $\mathbf{a}_{\text{loc}}^{t}$ ($\mathbf{a}_{\text{agg}}^{t}$) is computed normalizing the client deviations by

$$\mathbf{a}_{\iota}^{t}[k] \overset{def}{=} \frac{\mathbf{v}_{\iota}^{t}[k]}{\sum_{j\in\mathcal{K}^t} \mathbf{v}_{\iota}^{t}[j]}, \quad \forall k \in \mathcal{K}^t, \iota \in \{\text{loc, agg}\}. \quad (12)$$

The federated weight attention vector $\mathbf{a}^t$ is defined by

$$\mathbf{a}^{t}[k] \overset{def}{=} \begin{cases} \frac{n_k}{\sum_{j\in\mathcal{K}^t} n_j}, & \text{if } t = 0 \\ \frac{\mathbf{a}_{\text{agg}}^{t}[k] + \mathbf{a}_{\text{loc}}^{t}[k]}{2}, & \text{if } t > 0 \end{cases} \quad (13)$$

Intuitively, each $\mathbf{a}^t[k]$ represents a measure of client drift: as prototypes computed using weights $W_k \in \mathcal{W}_k$ of a model of a client k deviate from reference prototypes in terms of margin (either locally or on server), higher attention is applied on the weights $W_k \in \mathcal{W}_k$, and vice-versa. We remark that, according to our definition, if a client is not able to build reliable latent representations (low margin), then its model is considered less during aggregation.

Finally, federated attention vectors $\mathbf{a}^t$ are used to aggregate local weights at each t^{th} round by

$$W^{t+1} = \sum_{k\in\mathcal{K}^t} \mathbf{a}^t[k] W_k^t. \quad (14)$$

Our proposed FL method which employs (14) to solve (1) is called FedProto and is summarized in Algorithm 1.

Algorithm 1 FedProto.

Input: $\mathcal{K}, T, F, W^0, \eta, N$.
for $t = 0$ **to** $T - 1$ **do**
  A server samples $\mathcal{K}^t \subseteq \mathcal{K}$ clients $\propto p_k$, and sends W^t.
  for $k \in \mathcal{K}^t$ **do**
    Compute local prototypes (4).
    Update W_k^t with L_k (2) and step size η to W_k^{t+1}.
    Compute local prototypes (4) and LPM (8).
    Send W_k^{t+1}, LPM and $\hat{\mathbf{p}}_k^t$ back to the server.
  end for
  The server computes APM (9), $\mathbf{a}^t$ (13) and W^{t+1} (14).
end for

4. Experimental Setup

We evaluate on various tasks, models and real-world federated vision datasets. Full details on the experimental setup are given in Suppl. Mat. and are summarized in Table 1.

Classification Data. We evaluate FedProto on four classification datasets adopted from the related work [5, 20, 35]. First, we generate **synthetic** data following [31, 47], with addition of heterogeneity among clients. We sample from a logistic regression model with two parameters: ϕ_1, controlling how much local models differ from each other, and ϕ_2 controlling how much local data distribution at each client differs from that of other clients. To obtain highly non-i.i.d. data, we set $\phi_1 = \phi_2 = 1$ being the most heterogeneous, yet challenging, scenario. Other analyses have been carried out in [31]. Assuming that the data generation model is agnostic, we use a cascade of 2 dense and a softmax layer.

Then, we employ some real-world classification data. We distribute **MNIST** [28] data among $1,000$ clients such that each client has samples of only 2 digits (out of 10) and the number of samples per client follows a power-law [5]. We use the federated version of EMNIST [9, 5] (**FEMNIST**) proposed in [31] where 10 lower-case letters are subsampled and only 5 classes are distributed to each client. Finally, we generate non-i.i.d. **CelebA** [34] data (for smile classification), such that the underlying data distribution for each user is consistent with the raw data.

Image Segmentation Data. We use the VOC2012 [13] semantic segmentation dataset. We restrict to images with one single class inside (and the background) to mimic classification splits. We devise two class sets: macro (4 classes) and standard (20 classes). We distribute data to each client according to a Dirichlet distribution over the number of classes on each client with concentration parameter $\alpha > 0$, where low α values mean high non-i.i.d. data among clients, and vice-versa [19, 20]. The number of samples per client follows a power-law distribution with parameter $\gamma = 3$.

Implementation Details. Utilized hyper-parameters are reported on the right side of Table 1. We tuned learning parameters of each dataset on FedAvg (with $F = 1$ and no system heterogeneity) and, for fair comparison, we use the same parameters on all experiments for that dataset. We set $|\mathcal{K}^t| = 10, \forall t$ for all datasets. Randomly selected clients and mini-batch orders are kept fixed across all runs

Table 1. Statistics of the employed datasets (left) and hyper-parameters (right). In segmentation datasets, image background is excluded, and the accuracy refers to the mIoU. DeepLab-V3+ [8] uses MobileNet-V2 [18, 44] as the backbone pre-trained on ImageNet [27].

Dataset	# Classes	Clients	Samples	Samples/Client Mean	Samples/Client Std.	Model	Distribution	Central. Acc. (%)	Start lr	Solver	F	Rounds	Batch size
Synthetic	10	30	9,600	320.0	1051.6	2 dense layers	Power-law	78.5	0.01	SGD	20	200	10
MNIST	10	1,000	61,676	61.7	164.7	2-layer CNN	Power-law	99.0	0.01	SGD	20	200	10
FEMNIST	10	200	16,421	82.1	143.0	2-layer CNN	Power-law	99.0	0.001	SGD	20	400	10
CelebA	2	9343	177,457	19.0	7.0	4-layer CNN	Power-law	92.6	0.1	SGD	20	200	10
FPascal Macro	4	100	6,665	66.7	25.7	DeepLab-V3+	Power-law	79.7	10^{-4}	Adam	2	400	16
FPascal	20	100	6,665	66.7	25.7	DeepLab-V3+	Power-law	66.3	10^{-4}	Adam	2	400	16

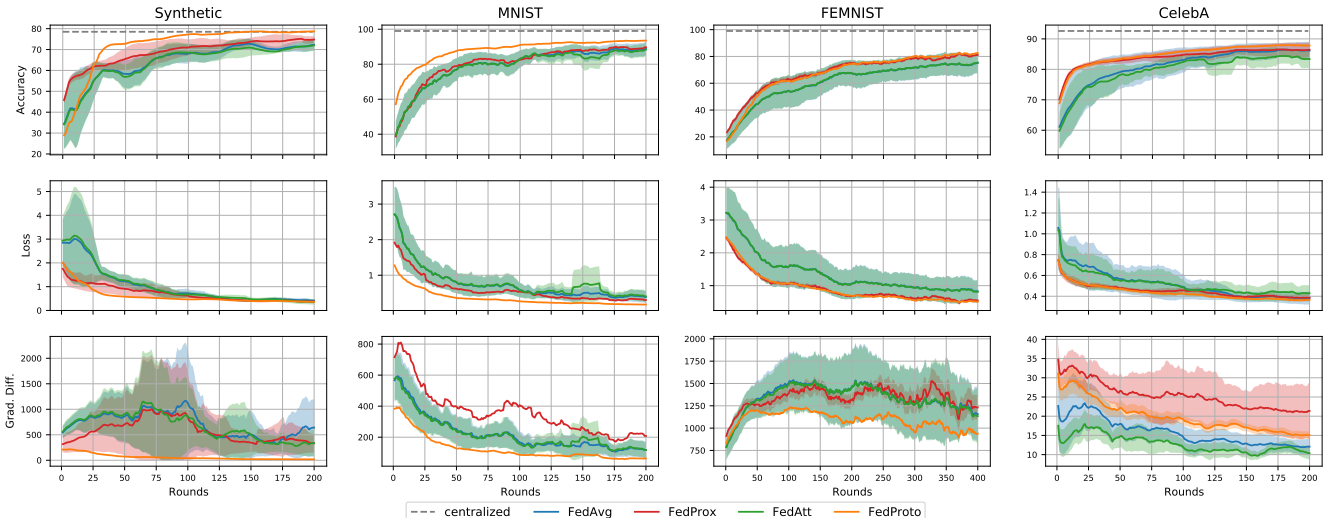

Figure 2. Experimental results for the classification task. Evaluation is performed across $\delta \in \{0\%, 50\%, 80\%\}$ and a moving average window of 10% rounds is applied for visualization. Solid lines and shaded regions represent the mean and standard deviation, respectively.

for comparative experiments. For simplicity, we use a constant learning rate on classification tasks, and polynomially decaying learning rate with power 0.9 and weight decay $4 \cdot 10^{-5}$ [7, 8] for segmentation tasks. For classification data, we measure accuracy as the percentage of correct predictions, whilst for segmentation data we use the mean Intersection over the Union (mIoU). All simulations are performed for 10% rounds more, and metrics are moving averaged over a window of 10% rounds in the visualization. The algorithms are implemented in Tensorflow [1] and are trained on a single NVIDIA RTX 2080Ti GPU.

5. Experimental Analyses for Federated Vision

5.1. Federated Image Classification

In this section, we report an extensive evaluation of our approach on image classification tasks. We compare FedProto with the baseline FedAvg, with the state-of-the-art regularization-based FedProx [31] and with FedAtt [22], which employs weight-based attention. Fig. 2 shows per-round aggregate accuracy, training loss and gradients difference on the four classification datasets introduced in Sec. 4. From the first row of Fig. 2, we observe that FedProto robustly outperforms FedAvg in terms of accuracy on every dataset. FedAtt brings minimal improvement compared to FedAvg, proving that a simple weight-based attentive mechanism is not very useful in vision tasks. FedProx, instead, leverages accuracy thanks to the toleration of partial workload and presence of the proximal term. However, our approach can effectively match or surpass the accuracy of FedProx. Additionally, we observe that both FedProto and FedProx show much lower variance (narrower shaded region) than competing approaches by tolerating partial results. Similar considerations are also reflected on the training loss (second row). The third row reports the average of squared ℓ_2 norm of difference of gradients over all clients, *i.e.*, $\frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} ||\nabla L_k(W^t; \mathcal{S}_k) - \nabla L(W^t)||_2^2$. As in [31], we interpret this dissimilarity measure as a proxy of accuracy. In particular, we observe how FedProto shows smaller dissimilarity (*i.e.*, better convergence [31]) compared to FedProx, thanks to the regularization effects brought by the proposed modules. To better appreciate accuracy and loss gaps observed in Fig. 2, we give results obtained using the aggregate models at the final round in Table 2 where FedProto shows significant improvements across all the datasets.

Ablation Studies. To explore the effect of the components of our approach on accuracy, we report a comparative ablation study in Table 3. First, we noticed that our approach tends to produce weights $\mathbf{a}^t$ deviating less from a fairness policy (*i.e.*, aggregation of weights W_k^t

Table 2. Final mean and std of accuracy (%) and loss from Fig. 2. Centralized accuracy are 78.5, 99.0, 99.4, 92.6, and losses are 0.33, 0.00, 0.00, 0.15 for Synth., MNIST, FEMNIST and CelebA.

		FedAvg	FedProx	FedAtt	FedProto
Accuracy	Synthetic	72.3 ± 2.6	74.8 ± 1.6	72.1 ± 2.7	$\mathbf{78.7 \pm 0.2}$
	MNIST	88.8 ± 3.8	$91.7 \pm \mathbf{0.2}$	88.4 ± 3.7	$\mathbf{93.3 \pm 0.2}$
	FEMNIST	75.1 ± 7.7	81.1 ± 1.0	75.5 ± 7.5	$\mathbf{82.5 \pm 0.3}$
	CelebA	86.2 ± 2.8	86.4 ± 2.4	83.4 ± 3.0	$\mathbf{87.8 \pm 0.4}$
Loss	Synthetic	0.41 ± 0.06	0.37 ± 0.07	$\mathbf{0.36} \pm 0.12$	$\mathbf{0.36 \pm 0.02}$
	MNIST	0.39 ± 0.17	$0.30 \pm \mathbf{0.02}$	0.41 ± 0.16	$\mathbf{0.18 \pm 0.02}$
	FEMNIST	0.83 ± 0.35	0.55 ± 0.04	0.81 ± 0.34	$\mathbf{0.51 \pm 0.01}$
	CelebA	0.38 ± 0.06	0.39 ± 0.03	0.43 ± 0.08	$\mathbf{0.36 \pm 0.02}$

Table 3. MNIST classification accuracy (%) of different strategies.

	δ			
Method	0%	50%	80%	Avg. $\pm$ Std.
FedAvg	92.7	88.7	85.1	88.8 ± 3.8
Fairness	92.8	89.9	86.5	89.7 ± 3.2
Fairness sampling i.i.d.	92.5	88.4	84.9	88.6 ± 3.8
FedAvg + toleration	92.7	90.2	89.1	90.7 ± 1.8
Fairness + toleration	92.8	91.2	90.6	91.5 ± 1.1
FedProto (no toleration)	**93.5**	90.8	88.1	90.8 ± 2.7
FedProto (d^+) only	92.8	92.4	92.1	92.4 ± 0.4
FedProto (APM only)	93.0	92.7	92.6	$92.8 \pm \mathbf{0.2}$
FedProto (LPM only)	91.9	91.0	90.6	91.2 ± 0.7
FedProto	**93.5**	**93.4**	**93.1**	$\mathbf{93.3 \pm 0.2}$

by $\mathbf{f}^t[k] = 1/K', \forall k, \forall t$) than FedAvg, as also observed in other contemporary approaches [57, 61]. A fair policy (row 2), indeed, outperforms FedAvg by a small margin. However, we argue that this is an implicit effect of the weighted sampling scheme of the active clients at each round introduced in Sec. 2. As a matter of fact, sampling active clients i.i.d. (row 3) brings results comparable to FedAvg. Second, we analyse the toleration of partial workload. Adding it on top of naïve implementations of FedAvg and Fairness (rows 4 and 5) improves the accuracy and the robustness over different amounts of δ. At the same time, we remark that our approach can achieve competitive performance even without tolerating partial workload (row 6).

Analyses of our model design are given in the last block of Table 3. Margin-based deviation can be viewed as an enhanced measure rather than just using the distance between prototypes belonging to the same class, *i.e.*, using only d^+ (row 7) from (5) to accommodate the class-wise probability distribution obtained from the distributed clients during aggregation. Although providing considerable improvements compared to FedAvg, we found margin-based deviation to be generally more stable. Finally, employing only one of the two proposed clues (LPM and APM in rows 8 and 9) still improves accuracy, and the combination of the two (last row) outperforms the effect of the singular components.

Aggregate Mean Margin (AMM). To examine margin maximization properties of federated optimizers, we define a measure called aggregate mean margin (AMM) by

$$\bar{\boldsymbol{\mu}}[t] = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mu(\mathbf{p}^t_{agg}[c], \mathbf{p}^t_{agg}). \tag{15}$$

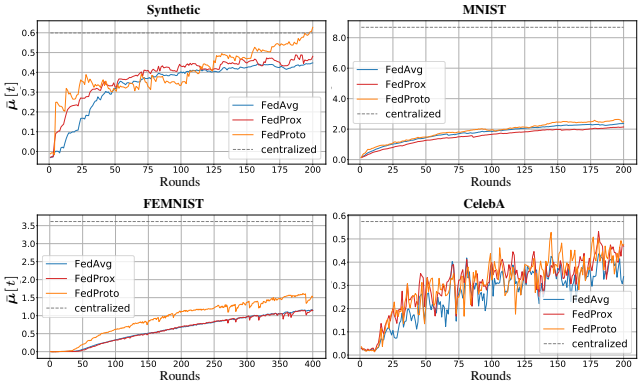

Figure 3. Per-round AMM ($\bar{\boldsymbol{\mu}}[t]$) values on classification datasets.

Table 4. Margin $\bar{\boldsymbol{\mu}}[T]$ of the final aggregate model and FFD (%).

		Synthetic	MNIST	FEMNIST	CelebA
$\bar{\boldsymbol{\mu}}[T]$	FedAvg	0.45	2.38	1.15	0.34
	FedProx	0.48	2.14	1.14	0.47
	FedProto	**0.63**	**2.49**	**1.51**	**0.48**
	Centralized	0.60	8.68	3.61	0.48
FFD	FedProx	61.9	5.9	1.5	4.5
	FedProto	**64.5**	**7.7**	**5.7**	**7.8**

In Fig. 3, we show change of AMM for different optimizers and datasets during training in FL. FedProto achieves higher $\bar{\boldsymbol{\mu}}[t]$ compared to other optimizers. This is a direct consequence of a better shaping of latent representations with improved class-discrimination acting as regularizer for learning meaningful feature representations similar to centralized training. The AMM for the last round, $\bar{\boldsymbol{\mu}}[T]$, is reported in Table 4. The results show a positive correlation between AMM and accuracy (given in Table 2) with Pearson's correlation coefficient $\rho = 0.68$ (p-value 0.01).

Federated Feature Discrepancy (FFD). FFD is devised to analyze how feature distributions provided by a model M_A trained with a federated optimizer A are closer to those generated by centralized training of a model M_C, compared to a baseline optimizer B. To this end, we first compute distribution $P^t_k, \forall k$ of features provided by M_A. Second, we train a model M_C on the same dataset without any distributed setting, and Q denotes the distributions obtained from M_C. Then, we compute the average Maximum Mean Discrepancy (MMD) [15] between P^t_k and Q by

$$MMD^t_A = \frac{1}{|\mathcal{K}^t|} \sum_{k \in \mathcal{K}^t} MMD(P^t_k, Q). \tag{16}$$

We define the FFD (%) between A and B as the relative gain of MMD^T_A over MMD^T_B by

$$FFD(A, B) = \frac{MMD^T_B - MMD^T_A}{MMD^T_B} \times 100. \tag{17}$$

Since our interest is to give a comparison with respect to the baseline FedAvg, we set A to FedProx or FedProto and

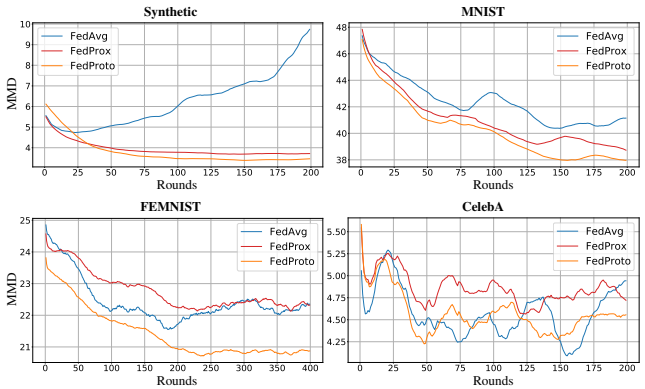

Figure 4. Per-round MMD (16) on classification datasets.

B to FedAvg. The per-round MMD is shown in Fig. 4 and the final FFD values (the higher the better) are reported in the bottom part of Table 4. Overall, we observe that distributions of features learned using FedProto are consistently more similar to those learned in centralized training than FedAvg. Last, we also note that FedProx can achieve some latent regularization thanks to the proximal term, however it is robustly surpassed by our proposed FedProto.

5.2. Federated Semantic Segmentation

We analyze our FedProto for federated semantic segmentation. Differently from image classification, segmentation task is more challenging as it involves dense predictions and highly class-imbalanced datasets. Altogether, these circumstances make aggregating local models even more severe.

We start by analyzing the effect of i.i.d. structure (i.i.d.-ness) of data on mIoU of federated segmentation models. For this purpose, we distribute two benchmark datasets among clients using the Dirichlet distribution with concentration parameter α (details are given in Sec. 4 and in Suppl. Mat.). Then, we train models on distributed data using the baseline FedAvg and our FedProto. The results depicted on Fig. 5 show the relationship between convergence of models and i.i.d.-ness of data. Note that, as the non-i.i.d.-ness of distributed data increases by lower α, data heterogeneity and client drift increase. Our FedProto improves mIoU and robustness compared to FedAvg on every configuration, and especially on highly non-i.i.d. data, where class-conditional representations on certain remote clients could be non-reliable due to the non-i.i.d. partitioning (only few samples for particular classes observed on certain clients).

A qualitative analysis on segmentation and entropy maps of two sample images comparing the final aggregate models of FedAvg and FedProto on different data splitting configurations (*i.e.*, setting $\alpha \in \{0.01, 0.1, 1\}$) is reported in Fig. 6.
Segmentation maps: Output segmentation maps (rows 1 and 4) improve when data are more i.i.d., better resembling segmentation maps produced by centralized training. FedProto significantly outperforms FedAvg for more non-i.i.d. data ($\alpha = 0.01$): the cat in first row and the horse in fourth row are correctly labeled and well-defined, whilst FedAvg labels them as a mixture of other animals. The ability to distinguish between class ambiguity is the direct consequence of a better latent space organization and regularization that FedProto achieves by maximizing prototype margin.

Entropy maps: Second, we report the entropy map of the softmax probabilities of the final model (rows 2 and 5): *i.e.*, $H_S = H(M(\mathcal{W}^T; \mathcal{X}))$, with $H(\cdot)$ being the pixel-wise Shannon entropy [52, 53]. Low entropy (dark blue) indicates a peaked distribution which is the reflection of high confidence of the network on its prediction, and vice-versa. Ideally, the entropy should be low for every pixel. However, as we can observe from centralized training, contours of objects and certain regions of the images (*e.g.*, the mane of the horse in row 4) have high entropy due to uncertainty on the precise edge localization of the objects or due to intrinsic ambiguity with other classes (all considered animal classes have fur with similar pattern). With these considerations in mind, we observe how FedProto produces generally darker entropy maps than FedAvg, especially on non-i.i.d. data. Last, we analyze the feature-level entropy maps upsampled to match input resolution (rows 3 and 6). To compute it, features $E(\mathcal{W}^T; \mathcal{X})$ are first normalized to $\hat{E}(\mathcal{W}^T; \mathcal{X})$, such that the sum over the channels at each low-resolution pixel location is 1 (*i.e.*, in order for them to be considered as probability vectors), and then we define $H_E = H(\hat{E}(\mathcal{W}^T; \mathcal{X}))$. In this case, H_E measures how representative a feature is at each pixel location. Ideally, features corresponding to the desired class should be well activated so that the decoder can discriminate between them and assign the correct label: this is the case of centralized training where features corresponding to (certain parts of) the object class are bright (*i.e.*, high entropy denoting many activated patterns). We observe that FedProto produces a feature-level entropy map which is more similar to centralized training than the map produced by FedAvg (particularly visible for low α values).

6. Conclusion

In this paper, we proposed FedProto, a distributed machine learning paradigm for vision models that can handle clients characterized by system and statistical heterogeneity. Previous approaches disregard internal representations to aggregate model weights. FedProto, instead, computes client deviations based on the inner class-conditional prototypical representations and uses them to drive federated optimization using an attentive mechanism. The experimental analyses demonstrated the effectiveness of our framework on both classification and segmentation datasets. In particular, we established a new benchmark on federated semantic segmentation task outlining a new research direction.

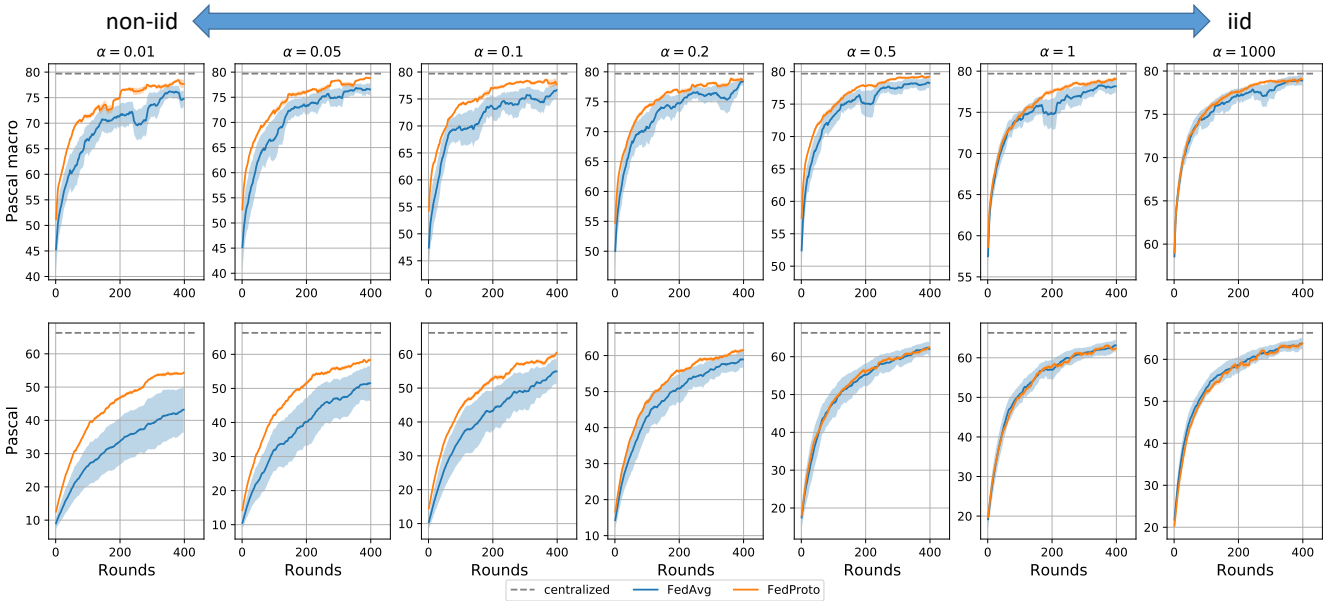

Figure 5. Change of mIoU on segmentation data distributed using different α values. Evaluation is performed across $\delta \in \{0\%, 50\%, 80\%\}$ and a moving average window of 10% rounds is applied. Solid and shaded lines represent mean and standard deviation.

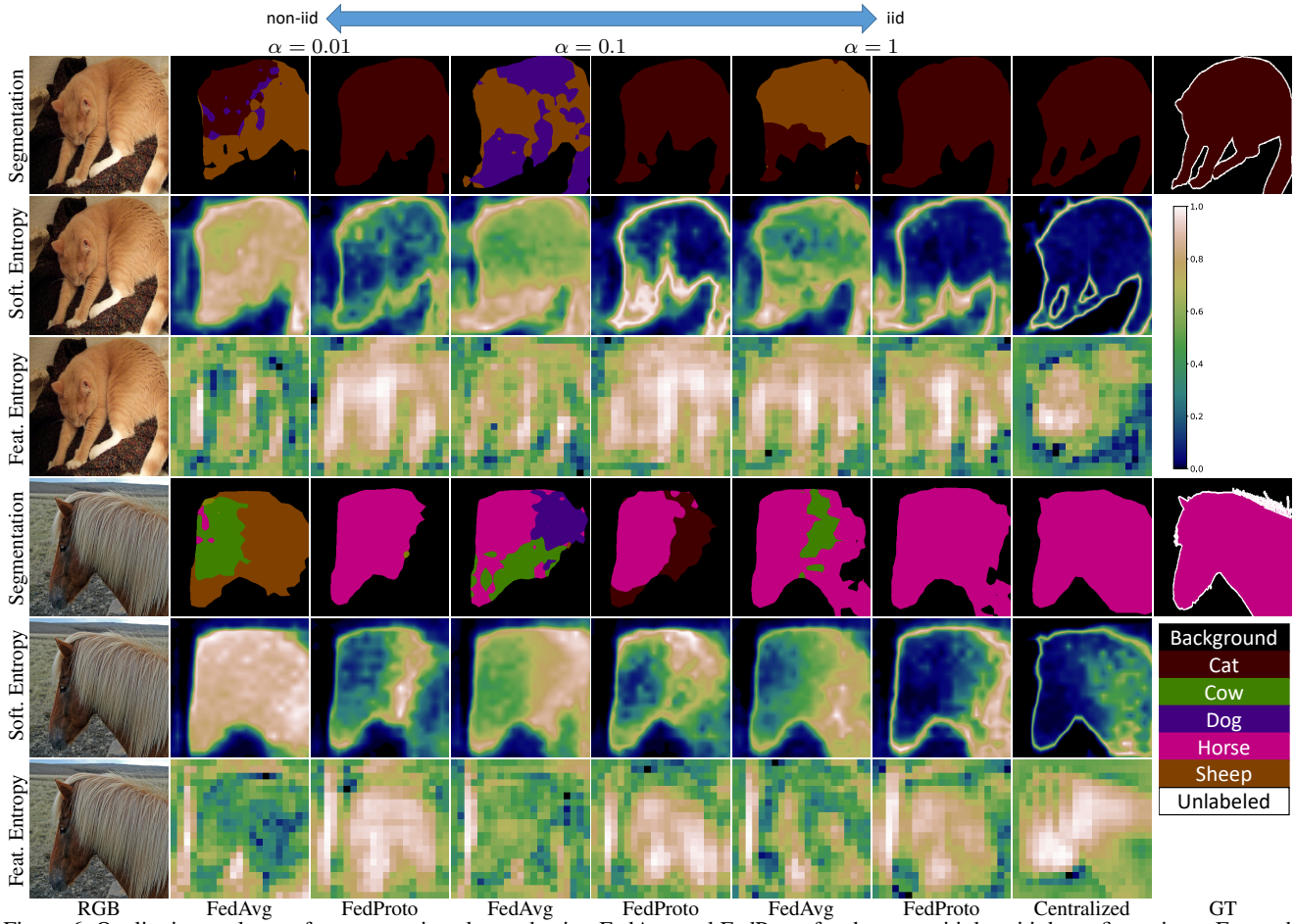

Figure 6. Qualitative analyses of representations learned using FedAvg and FedProto for three non-i.i.d. to i.i.d. configurations. For each of the two sample images, we show output segmentation maps (rows 1 and 4), softmax-level entropy maps (rows 2 and 5), entropy maps of internal features (rows 3 and 6). As reference, centralized training results are shown at the second last column. *Best viewed in colors*.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. 5, 2

[2] Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 232–241. PMLR, 2019. 2

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 35, pages 1798–1828. IEEE, 2013. 1

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *Conference of Machine Learning and Systems (MLSys*, 2019. 1, 3

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019. 4, 1

[6] Fabio Cermelli, Massimiliano Mancini, Yongqin Xian, Zeynep Akata, and Barbara Caputo. A few guidelines for incremental few-shot segmentation. *arXiv preprint arXiv:2012.01415*, 2020. 2

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 40, pages 834–848. IEEE, 2017. 5

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018. 5, 2

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017. 4

[10] Koby Crammer, Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin analysis of the lvq algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, pages 462–469, 2002. 2, 3

[11] Nanqing Dong and Eric P Xing. Few-shot semantic segmentation with prototype learning. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 3, 2018. 2, 3

[12] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: a Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 2

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. In *International Journal of Computer Vision*, volume 88, pages 303–338. Springer, 2010. 4, 1, 2

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. 1

[15] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 6

[16] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3973–3983. PMLR, 2020. 2

[17] Barbara Hammer, Marc Strickert, and Thomas Villmann. On the generalization ability of grlvq networks. *Neural Processing Letters*, 21(2):109–120, 2005. 2, 3

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5, 2

[19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019. 4, 2

[20] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020. 1, 2, 3, 4

[21] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized federated learning: An attentive collaboration approach. *arXiv preprint arXiv:2007.03797*, 2020. 2

[22] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 2, 5

[23] Xiao-Bo Jin, Cheng-Lin Liu, and Xinwen Hou. Regularized margin-based conditional log-likelihood loss for prototype learning. *Pattern Recognition*, 43(7):2428–2438, 2010. 2

[24] Peter Kairouz and H. Brendan McMahan. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14, 2021. 1, 3

[25] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5132–5143. PMLR, 2020. 2

[26] Junsik Kim, Tae-Hyun Oh, Seokju Lee, Fei Pan, and In So Kweon. Variational prototyping-encoder: One-shot learning with prototypical images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9462–9470, 2019. 2, 3

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 25:1097–1105, 2012. 5, 2

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4, 1

[29] Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12576–12584, 2020. 2, 3

[30] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. 1

[31] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Conference on Machine Learning and Systems (MLSys)*, 2020. 1, 2, 3, 4, 5

[32] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 2

[33] Xiaoqian Liu, Fengyu Zhou, Jin Liu, and Lianjie Jiang. Meta-learning based prototype-relation network for few-shot classification. *Neurocomputing*, 383:224–234, 2020. 2

[34] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the International Conference on Computer Vision (ICCV)*, December 2015. 4, 1

[35] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017. 1, 2, 4

[36] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent represetnations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

[37] David Nova and Pablo A Estévez. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3):511–524, 2014. 2, 3, 4

[38] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 719–729, 2018. 3

[39] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2239–2247, 2019. 2, 3

[40] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[41] Martin Rapp, Ramin Khalili, and Jörg Henkel. Distributed learning on heterogeneous resource-constrained devices. *arXiv preprint arXiv:2006.05403*, 2020. 1

[42] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020. 1

[43] Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1237–1244, 2003. 3

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 5

[45] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 423–429, 1995. 2, 3, 4

[46] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural computation*, 21(12):3532–3561, 2009. 2

[47] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1000–1008. PMLR, 2014. 4, 1

[48] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4080–4090, 2017. 2, 3

[49] Marco Toldo, Andrea Maracani, Umberto Michieli, and Pietro Zanuttigh. Unsupervised domain adaptation in semantic segmentation: A review. *Technologies*, 8(2), 2020. 2

[50] Marco Toldo, Umberto Michieli, and Pietro Zanuttigh. Unsupervised domain adaptation in semantic segmentation via orthogonal and clustered embeddings. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 1358–1368, 2021. 2

[51] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008. 4

[52] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7

[53] Weitao Wan, Jiansheng Chen, Tianpeng Li, Yiqing Huang, Jingqi Tian, Cheng Yu, and Youze Xue. Information entropy based feature pooling for convolutional neural networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3405–3414, 2019. 7

[54] Cong Wang, Yuanyuan Yang, and Pengzhan Zhou. Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity. *IEEE Transactions on Parallel and Distributed Systems*, 32(2):394–410, 2020. 1

[55] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 9197–9206, 2019. 2, 3

[56] Zbigniew Wojna, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings. The devil is in the decoder: Classification, regression and gans. In *International Journal of Computer Vision*, volume 127, pages 1694–1706. Springer, 2019. 2

[57] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting. *arXiv preprint arXiv:2012.00661*, 2020. 2, 6, 4

[58] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1426–1435, 2019. 3

[59] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019. 1

[60] Peihua Yu and Yunfeng Liu. Federated object detection: Optimizing object detection model with federated learning. In *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ICVISP 2019, New York, NY, USA, 2019. Association for Computing Machinery. 2

[61] Jingfeng Zhang, Cheng Li, Antonio Robles-Kelly, and Mohan Kankanhalli. Hierarchically fair federated learning. *arXiv preprint arXiv:2004.10386*, 2020. 6, 4

[62] Mengmi Zhang, Tao Wang, Joo Hwee Lim, Gabriel Kreiman, and Jiashi Feng. Variational prototype replays for continual learning. *arXiv preprint arXiv:1905.09447*, 2019. 2

[63] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018. 1

Prototype Guided Federated Learning of Visual Feature Representations
Supplementary Material

Umberto Michieli[1,2*] Mete Ozay[1]
[1]Samsung Research UK [2]University of Padova
{u.michieli, m.ozay}@samsung.com

In this document, we present supporting material about task definitions, model designs along with their respective hyper-parameters, and real-world federated datasets analysed in the main paper. Furthermore, we report some additional ablation study on image classification and some qualitative results on the semantic segmentation task.

S1. Federated Vision Datasets and Models

We evaluated our proposed FedProto on different computer vision tasks, models and real-world federated vision datasets. In this section, we explain the data generation process, data statistics and the models employed in our work. Most of our setups follow prior works [5, 31, 35].

S1.1. Synthetic Data Classification Dataset

First of all, we analyse our approach on highly non-i.i.d. synthetic data. For this purpose, we follow a similar setup to that proposed in [31, 47] with the addition of heterogeneity among clients.

- **Synthetic:** For each client k, we generate samples $(\mathbf{x}_k, y_k) \in \mathcal{X}_k \times \mathcal{Y}_k$ according to the logistic regression model $y_k = \arg\max_{c \in \mathcal{C}}(\text{softmax}(W\mathbf{x}_k + b))$, $\mathbf{x}_k \in \mathbb{R}^{60}$, $W \in \mathbb{R}^{10\times 60}$, $\mathbf{b}_k \in \mathbb{R}^{10}$. In the model, we first initialize $W_k \sim \mathcal{N}(u_k, 1)$, $\mathbf{b}_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, \phi_1)$, $\mathbf{x}_k \sim \mathcal{N}(\mathbf{v}_k, \mathbf{\Sigma})$, where the covariance matrix $\mathbf{\Sigma}$ is diagonal with $\mathbf{\Sigma}_{j,j} = j^{-1.2}$. Then, each element of $\mathbf{v}_k$ is drawn from $\mathcal{N}(B_k, 1)$, $B_k \sim \mathcal{N}(0, \phi_2)$. Hence, ϕ_1 controls how much local models differ from each other, ϕ_2 controls how much local data distribution at each client differs from that of other clients. For our simulations, we set $\phi_1 = \phi_2 = 1$ being the most heterogeneous, yet challenging, scenario. Other analyses have been carried out in [31]. Assuming that the underlying data generation model is agnostic, we employ a cascade of 2 dense layers with 128 and 256 units respectively, followed by a softmax output layer.

S1.2. Real World Image Classification Datasets

To further investigate accuracy on classification data, we explore real-world image classification datasets, inspired from [5, 31, 35].

- **MNIST:** It is a classification task of $28 \times 28px$ images containing handwritten digits 0-9 [28]. To simulate a heterogeneous (non-i.i.d.) setting, we distribute data among $1,000$ clients such that each client has samples of only two digits and the number of samples per client follow a power law distribution. To tackle this task we employ a simple custom network with 5×5 convolution layers (the first with 32 channels, the second with 64, each followed by 2×2 max pooling), a fully connected layer with 256 units and ReLu activation, and a final softmax output layer.
- **FEMNIST:** It is a classification task of $28 \times 28px$ images containing 62-class handwritten character digits [28]. Following [31], to generate heterogeneity we first subsample 10 lower case characters (from 'a' to 'j') and we distribute only 5 classes to each client. The number of clients is 200. To tackle this task we employ a simple custom network with 5×5 convolution layers (the first with 32 channels, the second with 64, each followed by 2×2 max pooling), a fully connected layer with 256 units and ReLu activation, and a final softmax output layer.
- **CelebA:** Finally, we generate non-i.i.d. CelebA [34] data (for classification of smiling faces), such that the underlying distribution of data for each user is consistent with the raw data. For this task, we use a 4-layer CNN each with 32 channels and followed by 2×2 max pooling and ReLU activation, and a softmax layer.

S1.3. Real World Semantic Segmentation Datasets

Then, we investigate the accuracy on a dense prediction task, such as semantic segmentation. We propose a new benchmark for federated semantic segmentation employing the Pascal VOC2012 dataset [13] in two different flavours.

- **Pascal:** We use the standard Pascal VOC2012 [13] semantic segmentation dataset. We consider only images with one single class inside (in addition to the background) in order to mimic classification splits. There are a total of 20

*Researched during internship at Samsung Research UK.

object-level classes (background excluded) and we distribute data to each client according to a Dirichlet distribution with concentration parameter $\alpha > 0$. Low values of α mean that dataset is highly non-i.i.d. among clients, and vice-versa for high α values [19, 20]. The number of samples per client follow a power-law distribution with parameter $\gamma = 3$. For this task, we use a DeepLab-V3+ [8] architecture with MobileNet [18] as the encoder pre-trained on ImageNet [27].
To further elucidate on the data splitting mechanism, we report some dataset statistics for different values of α in Figure S1. The first column represents the distribution of the number of classes present on clients, while the second column represents the distribution of clients having a certain amount of classes. In the most non-i.i.d. case considered (*i.e.*, $\alpha = 0.01$), each client only experiences samples from a few classes, while they progressively observe more and more classes as the i.i.d.-ness improves. The third column of Figure S1 illustrates populations drawn from the Dirichlet distribution with different concentration parameters. For visualization purposes, we restrict to 30 randomly sampled clients and each color refers to a different class (color coding scheme reflects Pascal VOC2012 colormap). As expected, for low values of α, the distributions are similar but not identical to a sort-and-partition approach in which each client only sees samples of one class (plus the background) [19], since we have a highly imbalanced number of samples per each class and samples are distributed to clients according to a power-law distribution. Experimentally, we verified that a simple extremely i.i.d. sort-and-partition approach yields same results as the case with $\alpha = 0.01$.

- **Pascal macro:** We follow the same exact splitting described for standard Pascal VOC2012 with the only difference that classes are hierarchically grouped according to their semantic meaning into 5 classes (background included). The coarser set of classes is derived from the notional taxonomy from [13, 12]. The map from 21 to 5 classes is:
  - Background: *Background*;
  - Person: *Person*;
  - Vehicles: *Aeroplane*, *Bicycle*, *Boat*, *Bus*, *Car*, *Motorbike*, Train;
  - Household: *Bottle*, *Chair*, *Dining Table*, *Potted Plant*, *Sofa*, *TV/Monitor*;
  - Animals: *Bird*, *Cat*, *Cow*, *Dog*, *Horse*, *Sheep*.

S2. Hyperparameters and Implementation Details

For all the considered datasets, we randomly split the data on each local client into a training and a testing set with a $80/20$ ratio. We fix the number of selected clients to be 10 for all experiments and most of the hyper-parameters has been reported in Table 1 of the main paper. Unless otherwise stated, we assume that FedAvg does not tolerate partial local solutions (*i.e.*, dropped clients are not aggregated), while FedProto and FedProx do tolerate them. For Synthetic, MNIST, FEMNIST, and CelebA, we set the proximal loss term of FedProx following the guidelines of [31], and the best accuracy is obtained respectively for: 0.1, 1, 1, 0.01.

We developed our framework in Tensorflow [1]. We simulate the federated learning setup (1 server and $|\mathcal{K}|$ clients) on a single NVIDIA GeForce RTX 2080 Ti GPU with 2 Intel Xeon Gold 5220 CPU at 2.20GHz.

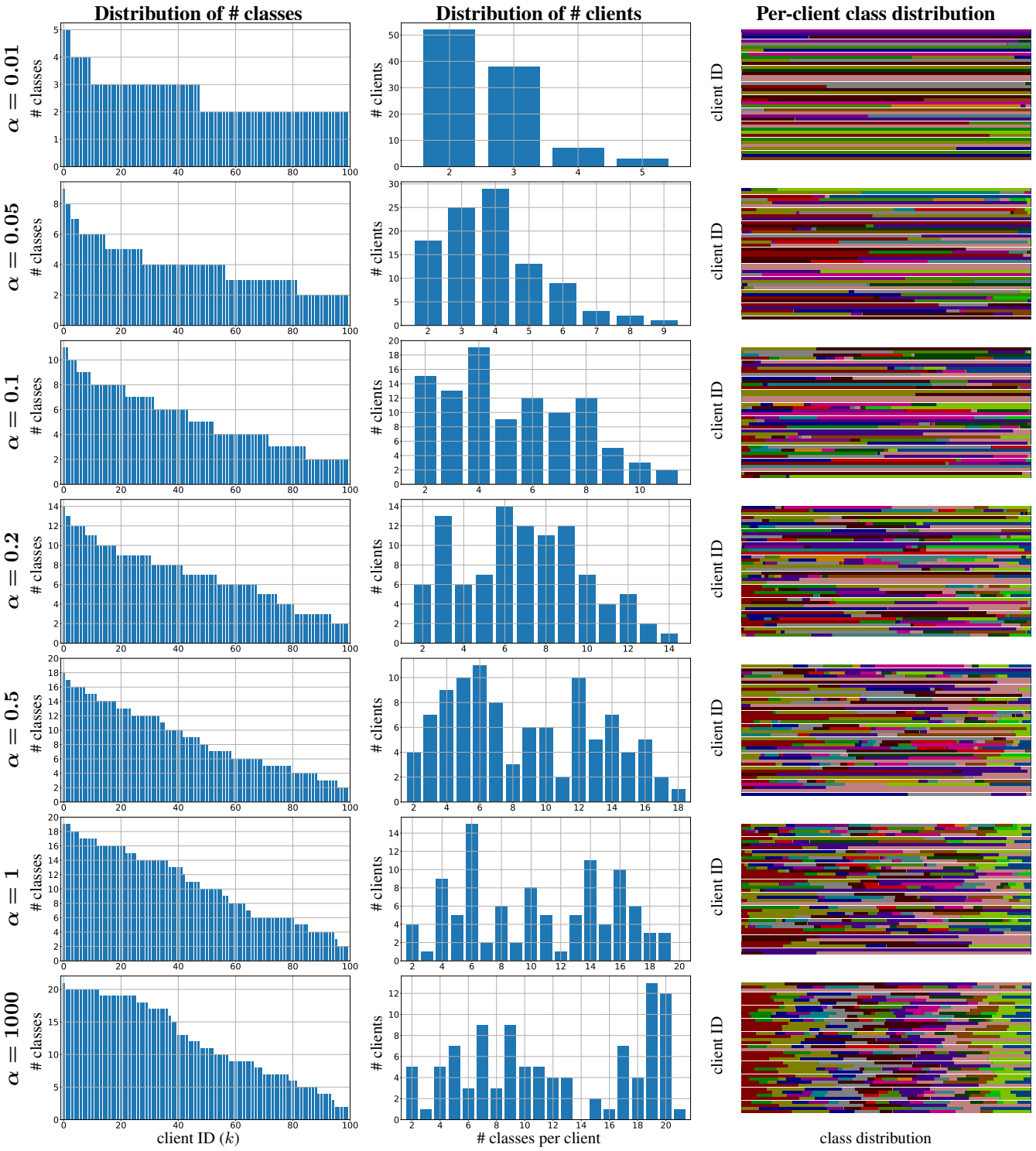

Figure S1. Dataset statistics of different data splitting schemes used by clients for the Pascal VOC2012 segmentation task. The first column reports the distribution of the number of classes among clients (note that the background is present in all the images). The second column shows the distribution of number of clients according to number of classes per client. The third column reports the per-client distribution of classes depicted with different colors, where the client IDs are restricted to 30 randomly sampled clients for visualization purposes (the background is not included in the visualization and the colors refer to the Pascal VOC2012 colormap). *Best viewed in colors.*

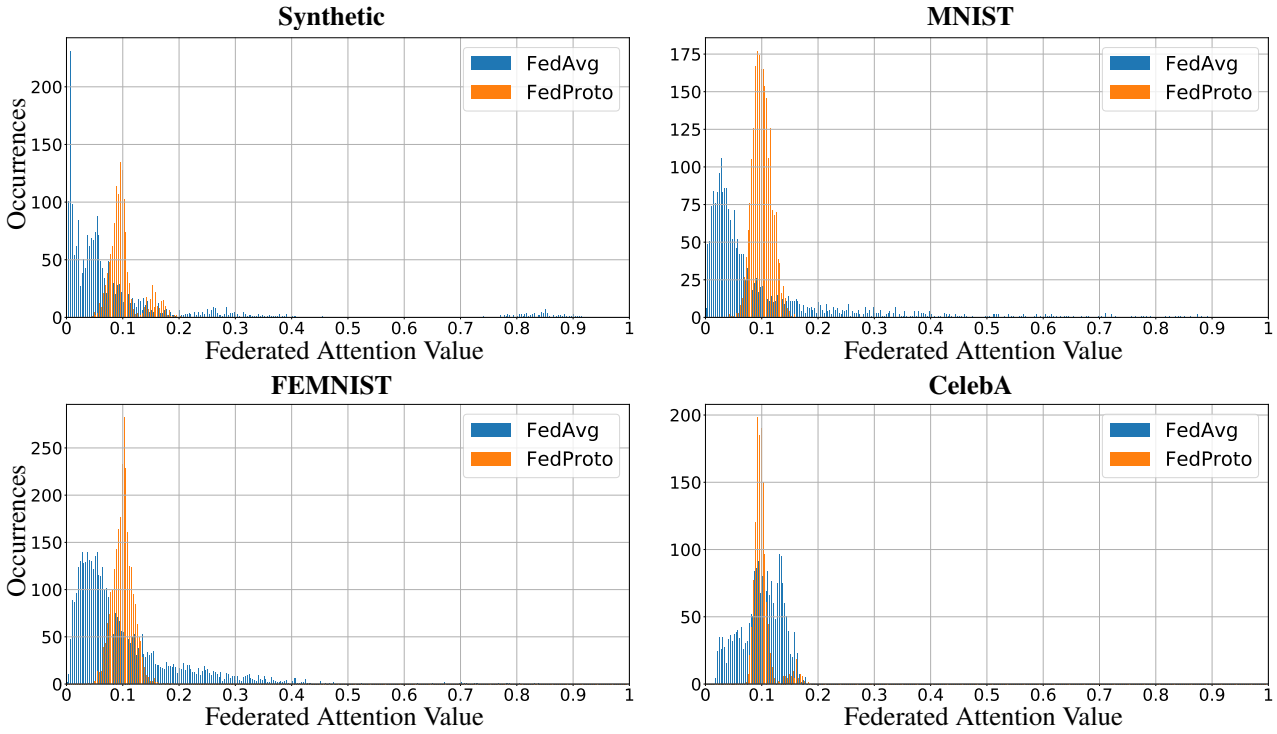

Figure S2. Comparison of distributions of the federated attention vector $\mathbf{a}^t[k], \forall k, \forall t$, on classification datasets for FedAvg and our FedProto.

S3. Additional Experimental Analyses

S3.1. Federated Attention Values in Image Classification

In the main paper, we mentioned that our approach tends to produce federated attention values $\mathbf{a}^t[k]$ deviating less from a fairness policy (*i.e.*, aggregation of weights W_k^t by $\mathbf{f}^t[k] = 1/K', \forall k, \forall t$) than FedAvg, as also observed in other contemporary approaches [57, 61]. In Figure S2, we compare the distribution of federated attention values $\mathbf{a}^t[k], \forall k, \forall t$, of FedAvg and of our approach. The results show that FedProto produces attention values having a much lower variance from the average value (we remark that $|\mathcal{K}| = 10$ is used) compared to FedAvg. In particular, FedAvg weights follow the same distribution of the number of samples, which could lead the framework to ignore clients with less samples during aggregation, regardless of the statistical distribution of local samples. Moving from these considerations, in the main paper we reported some results employing a fairness policy, which we found to have comparable results to FedAvg and to be significantly surpassed by our approach.

S3.2. Additional Qualitative Results on Semantic Segmentation

One of the main effects of our proposed FedProto is a class-conditional latent-level regularization, achieved via prototype guided federated optimization and margin maximization of the aggregate model during its distributed training. In order to visually represent the main effects, we report in Figure S3 the 2D t-SNE embeddings of the features of the final aggregate model [51] for different values of α. Here, the background class is not included and the colors refer to the Pascal VOC2012 colormap. Class membership for the low-resolution feature map is obtained with simple nearest neighbor downsampling of the full-resolution segmentation maps. By visual inspection, we observe that t-SNE embeddings produced using the final aggregate model from FedProto are better subdivided into clusters (*i.e.*, points of the same color). In particular, for $\alpha = 0.01$, FedAvg confuses some animal classes (horse in pink, sheep in brown, cow in green, cat in dark red and dog in purple) into one mixed point cloud on the top right part of the plot, lacking class-discrimination at the feature level, as argued in the main paper. FedProto, instead, produces a much clearer separation among these (and others) classes, being able to build class-discriminative clusters at the latent level (*i.e.*, clusters points on the basis of their class membership). Similar discussion can be made also for the remaining scenarios, with a progressively smaller difference between t-SNE embeddings produced by FedAvg and FedProto as the data i.i.d.-ness increases.

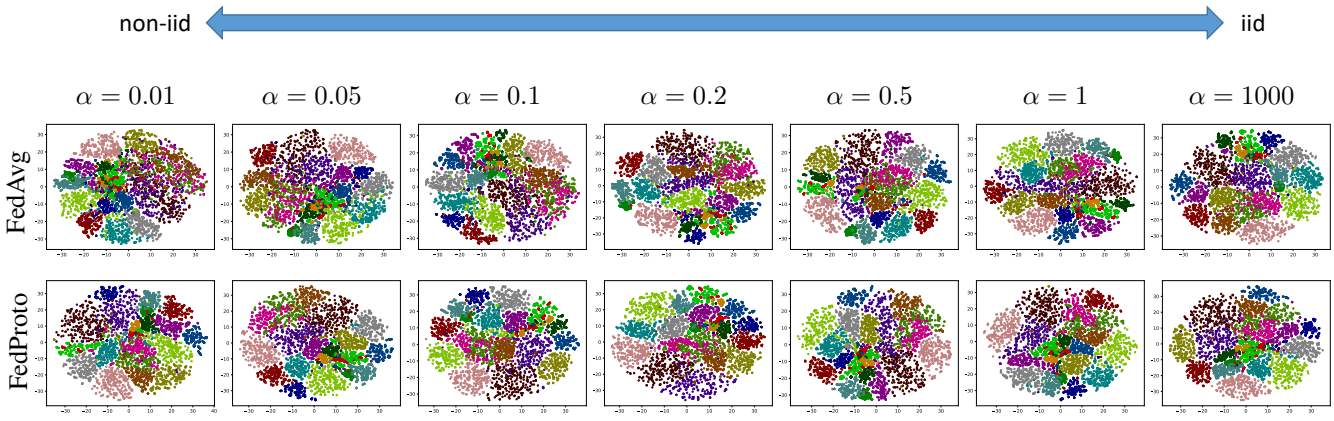

Figure S3. Comparison of t-SNE embedding plots of feature representations learned by FedAvg and by our FedProto, using the Pascal VOC 2012 segmentation benchmark with 20 object level classes. Analyses are performed over different values of α. The *background* class is not included in the visualization, and the colors refer to the Pascal VOC2012 colormap. *Best viewed in colors.*

To conclude, we report in Figure S4 a qualitative analysis on segmentation and entropy maps of three sample images comparing the final aggregate models of FedAvg and FedProto on different data splitting configurations (*i.e.*, for different values of $\alpha \in \{0.01, 0.1, 1\}$). In particular, for each image, we show the predicted segmentation map (rows 1, 4 and 7), the entropy map of the final softmax layer (rows 2, 5 and 8) and the entropy map of the intermediate features (rows 3, 6 and 9). Looking at the overall picture, we can appreciate a general improvement when going from more non-i.i.d. to more i.i.d. data, as the complexity of the optimization decreases. Analogous considerations to those reported in the main paper also hold in this case.

Our model generally achieves higher quality **segmentation maps**, which generally improve when data are more i.i.d., better resembling segmentation maps produced by centralized training. FedProto produces better segmentation maps for more non-i.i.d. data compared to FedAvg: more correct class identification in the first sample (horse, instead of cow or sheep) and of the third sample (in the non-i.i.d. case), and better objects shaping in the last two sample images. The ability to distinguish between class ambiguity is the direct consequence of a better latent space organization and regularization that FedProto achieves by maximizing prototype margin.

Furthermore, FedProto provides less uncertainty on the chosen classification labels than FedAvg, as shown by the **softmax-level entropy maps**. Here, pixel-wise uncertainty on the prediction of the network is measured via entropy levels: the lower the entropy (*i.e.*, the darker the pixels) and the higher is the confidence of the prediction, being representative of a peaked distribution over the class probabilities.

Finally, we report the **feature-level entropy maps**, which measures how representative a feature is at each pixel location. Features corresponding to the desired class should be well activated so that the decoder can discriminate between them and assign the correct label: this is the case of centralized training, where features corresponding to (certain parts of) the object class are bright (*i.e.*, high entropy denoting many activated patterns). Also in this case, we observe that FedProto produces feature-level entropy maps more similar to centralized training than the maps produced by FedAvg (particularly visible for low values of α).

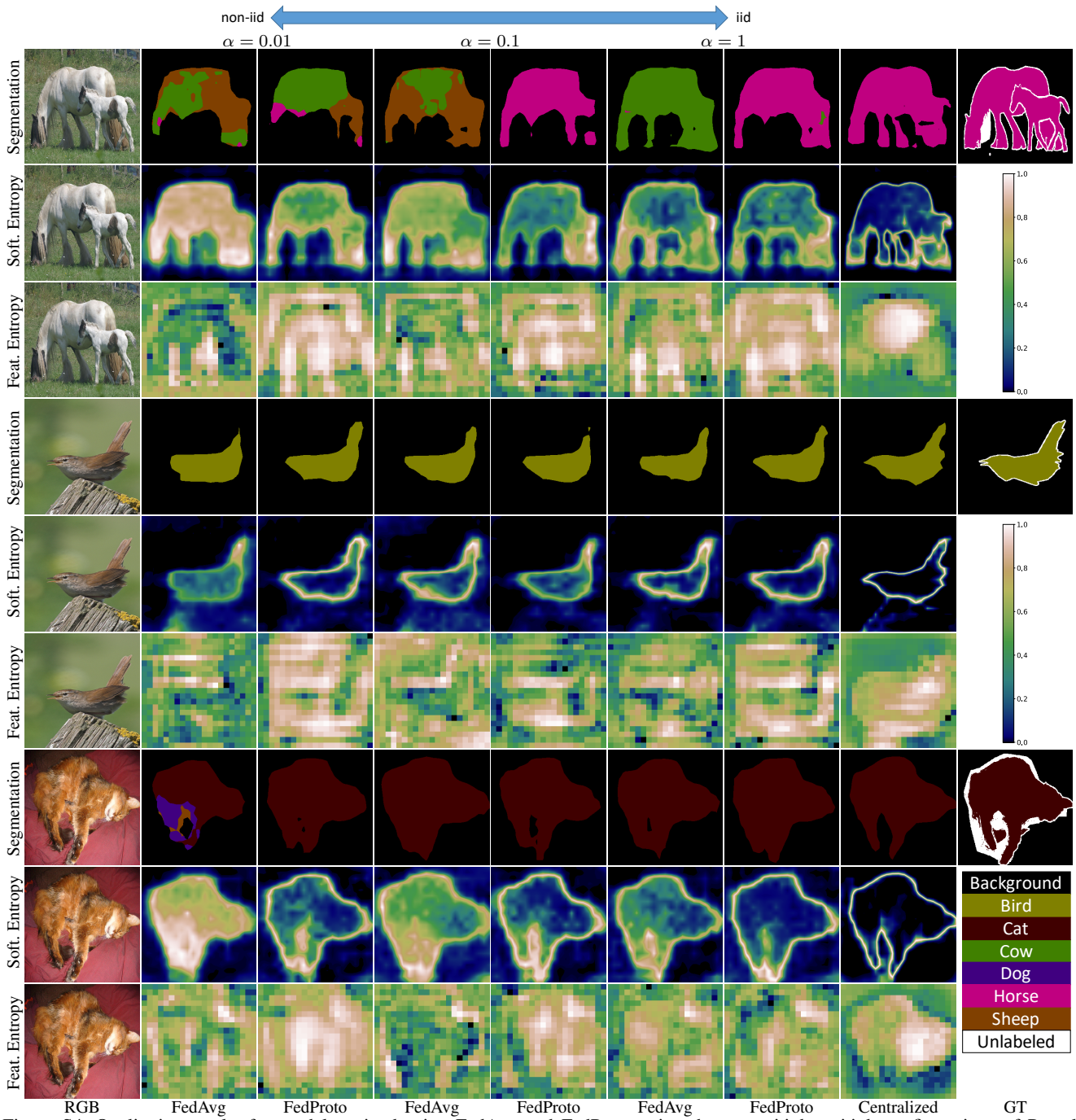

Figure S4. Qualitative results for models trained using FedAvg and FedProto using three non-i.i.d. to i.i.d. configurations of Pascal VOC2012 dataset. For each of the three sample images, we depict; the output segmentation map (rows 1, 4 and 7), the softmax-level entropy map (rows 2, 5 and 8), and the feature-level entropy map (rows 3, 6 and 9). As a reference, output maps of models obtained using centralized training are shown on the second last column. *Best viewed in colors*.