Cluster-driven Graph Federated Learning over Multiple Domains

Debora Caldarola[*,1], Massimiliano Mancini[2], Fabio Galasso[3],
Marco Ciccone[4], Emanuele Rodolà[3], Barbara Caputo[1,5]

[1] Politecnico di Torino, [2] University of Tübingen, [3] Sapienza University of Rome,
[4] Politecnico di Milano, [5] Italian Institute of Technology

Abstract

*Federated Learning (FL) deals with learning a central model (*i.e. *the server) in privacy-constrained scenarios, where data are stored on multiple devices (*i.e. *the clients). The central model has no direct access to the data, but only to the updates of the parameters computed locally by each client. This raises a problem, known as statistical heterogeneity, because the clients may have different data distributions (*i.e. *domains). This is only partly alleviated by clustering the clients. Clustering may reduce heterogeneity by identifying the domains, but it deprives each cluster model of the data and supervision of others.*

Here we propose a novel Cluster-driven Graph Federated Learning (FedCG). In FedCG, clustering serves to address statistical heterogeneity, while Graph Convolutional Networks (GCNs) enable sharing knowledge across them. FedCG: ***i.*** *identifies the domains via an FL-compliant clustering and instantiates domain-specific modules (residual branches) for each domain;* ***ii.*** *connects the domain-specific modules through a GCN at training to learn the interactions among domains and share knowledge; and* ***iii.*** *learns to cluster unsupervised via teacher-student classifier-training iterations and to address novel unseen test domains via their domain soft-assignment scores. Thanks to the unique interplay of GCN over clusters, FedCG achieves the state-of-the-art on multiple FL benchmarks.*

1. Introduction

In Federated Learning (FL) [30], a central server model is trained using data stored locally on multiple client devices. Each client computes a local update of the model and all the client updates are then aggregated server-side to build the final model. Since no data ever leaves the client devices, the central model has no direct access to the raw data itself, a fundamental requirement for privacy-preserving applications (e.g. medical records, bank transactions, etc.). FL usually relies upon the key assumption that a single central model can work efficiently across several users [15, 22]. This may not hold in practice, since distinct clients might hold different input distributions, *i.e. domains* (*e.g.*, people speaking different languages, pictures taken at different locations), with their data possibly being not identically distributed and/or unbalanced. These issues (collectively referred to as *statistical heterogeneity* [41]) imply that all i.i.d.

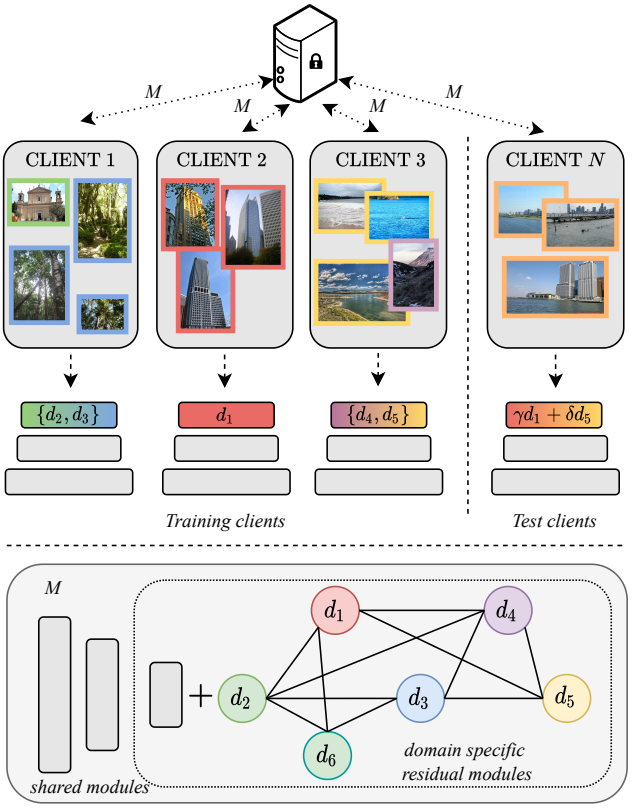

Figure 1. In a federated scenario, clients and server exchange the parameters of the model M. Each client has access to its local data, which can be non-i.i.d. and unbalanced. In the image, each color identifies a different distribution, *i.e.* a domain, such as pictures of skyscrapers or sea landscapes. Our model M is made of domain-agnostic layers (in gray) and a GCN containing domain-specific parameters, added as residual. According to the domains of the input images, the corresponding nodes of the GCN are activated. At test time, new domains can be addressed as a soft combination of the discovered ones, *e.g.* skyscrapers over the sea.

[*]Corresponding author: debora.caldarola@polito.it

assumptions made in distributed optimization or centralized training are violated, and modeling the learning problem requires taking into account this new complexity.

To the best of our knowledge, statistical heterogeneity has been tackled so far with diverse approaches, but none of them has modeled the direct share of knowledge between domains. In particular, meta-learning FL techniques focused on the client-server relation [14, 17, 8, 5]; multitask FL methods specialized parts of the models to certain clients [41, 7]; while clustering-based FL split the clients and data, learning separate models for them [40, 44].

In this work, we introduce a novel Cluster-driven Graph Federated Learning (FedCG). FedCG leverages clustering and its potential to reduce statistical heterogeneity by identifying homogeneous[1]. Concurrently, FedCG is the first to model the domain-domain interaction by means of a GCN, which connects domain-specific model components. In the GCN, each node consists of domain-specific model parameters, while the adjacency matrix is composed of the inverse pairwise distances between the domain-specific parameters. In this way, FedCG not only captures the specificity of each domain but also allows each domain to benefit from the updates of others, sharing knowledge at training.

Our clustering is based on unsupervised teacher-student [10] classifier-training iterations and it generalizes to unseen test-time domains. We cluster by pseudo-labels, assigned by a teacher and learned by a student, in rounds of refinements. This is accomplished within the FL training paradigm, respecting the client's privacy. This allows to estimate soft-assignments for unseen novel test domains.

We test our model extensively on several FL benchmarks, demonstrating results above or competitive with the state-of-the-art. Our main contributions are:

1. We present the first cluster-driven GCN-based approach to address statistical heterogeneity in the FL scenario. Thanks to the interactions among domains learned by the means of a GCN, knowledge is shared across domains according to a similarity-based criterion, reducing the risk of overfitting and helping the less populated domains.

2. We introduce an iterative teacher-student clustering algorithm designed for the federated learning scenario, which allows adapting to new domains via soft-assignments. This captures the diverse domain distributions without violating the FL constraints. Each domain is assigned model-specific components, trained via GCN interactions.

3. We evaluate our model on multiple FL benchmarks, where we compare favorably or on par with respect to the state-of-the-art.

[1]Homogeneous stands in this context for groupings that minimize intra-cluster Vs inter-cluster variance.

2. Related Work

Although FL is a relatively new field of study, it has aroused great interest in the research community because of its wide applicability in privacy-constrained scenarios [22]. A simple but effective baseline for FL is the Federated Averaging (FedAvg) algorithm [30], where the central model is obtained as a weighted average of the models received from each client after their local updates. FedAvg has been extensively studied and extended by changing either what is averaged, or how the local models are considered in each update. For instance, FedSGD [30] bases the update on the model's gradient instead of the weights, while in [19], the updates are parametrized with fewer variables to reduce the uplink communication cost. Similarly, in [38] the nodes only send a quantized version of their local information for reducing the communication overhead. Mohri *et al*. [32] propose Agnostic Federated Learning (AFL) in order to reduce the bias towards specific clients. In [21], each client designs its own local model and the local information is shared by means of knowledge distillation. Our work mainly relates to [30], but we explicitly revise the FedAvg framework to account for statistical heterogeneity.

Statistical heterogeneity in FL Despite their effectiveness, the previous methods ignore an important problem of FL, *i.e*. statistical heterogeneity. Many works [23, 11, 24, 16] study this challenge in terms of convergence analysis and effects of non-i.i.d. data distributions in the federated scenario. Others address this problem from the meta-learning [33] and multitask [4] perspectives for building specialized models. Specifically, Model-Agnostic Meta-Learning [9] caught the interest of the FL community for its compatibility with any ML model trained with gradient descent [14, 17, 8, 5]. On the other hand, in Federated Multi-Task Learning (FMTL) [41, 7], each client is seen as a different task. FMTL addresses underlying similarities and structures common to some clients by learning a separate model for each device of the network. In particular, [40] and [44] cluster clients according to their data distribution, assigning a specialized model to each cluster: [40] uses the cosine similarity of the clients' gradient update, while [44] dynamically groups clients exploiting structural similarities. Hsu *et al*. [12] develop FedIR and FedVC for re-sampling and re-weighting the client pools.

As in the aforementioned algorithms, in this work, we focus on addressing statistical heterogeneity in FL. Similarly to [40, 44], we seek to learn specialized models addressing the different data distributions. However, differently from [40], our clusters are built and updated *during* the federated communication rounds through a domain classifier, and not offline after the model convergence. This removes the requirement that all client-specific models must be stored server-side. Moreover, our method aims to identify and group distributions rather than the clients them-

selves, allowing us to model the realistic case where a client's data may belong to multiple distributions. In addition, while the applicability of [44] is constrained to the clients seen during training, our domain model can be applied to unseen clients at test time thanks to the flexibility of our domain classifier. Finally, we explore merging FL with Graph Representation Learning to address statistical heterogeneity, using graphs to learn domain-specific parameters and to model the interactions among them.

Graph Representation Learning Our work employs Graph Convolutional Networks [18], using domain-specific parameters as high-dimensional features at each node of the graph. This is partly inspired from [28], where domain-specific batch normalization [13] layers are connected through a graph for addressing predictive domain adaptation. However, here we focus on a completely different problem (*i.e.* FL), requiring a different training paradigm, and we build our graph on arbitrary layers of the network.

In the context of FL, to the best of our knowledge, the only existing works adopting graphs as auxiliary representations are SGNN [31], ASFGNN [47] and GraphFL [43]. The first two employ graphs for a different purpose: they use a similarity-based graph neural network for improving node classification in network embeddings, while preserving user privacy. GraphFL, instead, is a semi-supervised node classification method on graphs and uses the FL scenario to solve real-world graph-based problems. Differently from these works, we use the graph-based formulation not to learn a single general model, but to capture the statistical heterogeneity while taking into account the relations among the different data distributions. Thanks to the graph, each domain can be addressed with specific parameters while still taking advantage of all local updates.

3. Cluster-driven Graph Federated Learning

In this section we present our approach addressing statistical heterogeneity in FL by means of GCNs. Our method is based on three ideas: i) identify the clusters of data sharing the same distribution, ii) assign specific network components to each cluster, and iii) let the components interact within a GCN. We name our full model Cluster-driven Graph Federated Learning (FedCG). Before describing FedCG, in the following we formalize the FL problem.

Problem Formulation. Our goal is to learn a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, parametrized by θ, mapping samples from an input space $\mathcal{X}$ to their corresponding semantic in an output space $\mathcal{Y}$. Specifically, we focus on a classification task, where $\mathcal{X}$ contains images while $\mathcal{Y}$ is a probability simplex defined over a set of labels Y.

In the FL setting, the server does not have direct access to the data, but can communicate with a set C of clients, where each client $c \in C$ has access to a local dataset $\mathcal{T}_c = \{x_i, y_i\}_{i=1}^{n_c}$ with $x \in \mathcal{X}$ and $y \in Y$. In this scenario, we can learn f_θ by querying clients and relying on their local updates of the parameters θ. In particular, since $|C|$ is large, we can assume a synchronous update scheme proceeding in communication rounds, where in each round a set K of clients receives f_θ, with $|K| \ll |C|$. Each client $k \in K$ computes a local update of θ, *i.e.* θ_k, with its local dataset $\mathcal{T}_k$, by minimizing a given objective function. Since we consider classification tasks, we update θ_k by minimizing the standard cross-entropy loss over $\mathcal{T}_k$:

$$\theta_k = \min_\theta -\frac{1}{n_k} \sum_{(x,y)\in\mathcal{T}_k} \log f_\theta^y(x)\,, \tag{1}$$

where $f_\theta^y(x)$ denotes the probability of x to belong to class y as given by f_θ.

With Eq. (1), we obtain for each client $k \in K$ its corresponding local parameters θ_k tuned to address the classification task of $\mathcal{T}_k$. At each round, the server gathers all local updates and combines them to update the central model parameters θ. A simple yet effective strategy to aggregate the local updates is FedAvg [30], that computes θ as the weighted average of each θ^k:

$$\theta = \frac{1}{\sum_{k\in K} n_k} \sum_{k\in K} n_k \theta_k\,. \tag{2}$$

Heterogeneity may be a problem in FedAvg, and in general for FL strategies, due to the lack of convergence guarantees in non-i.i.d. and unbalanced data [41, 22]. In realistic applications, the joint probability distributions over $\mathcal{X}$ and Y are usually different in each client, *i.e.* given two clients c and k with $c \neq k$, we have $p_{\mathcal{X}Y}(\mathcal{T}_c) \neq p_{\mathcal{X}Y}(\mathcal{T}_k)$.

To address this problem, we propose an approach that i) identifies the distributions (*i.e.* domains) present in different clients through clustering; ii) instantiates domain-specific components to adapt the model to each domain; iii) makes the various domain-specific modules interact through a GCN, such that updating one of them can benefit the others. In the following we describe each of these elements.

3.1. Federated Clustering

To address statistical heterogeneity through domain-specific modules, we need to identify the different domains present in the data. This is challenging since data are split across multiple clients and the server cannot cluster them directly. Moreover, these clusters, even if correctly identified for the training set, may not be optimal for the test set. Here we address the first problem by a clustering procedure built on two *domain classifiers*, one having the role of the *teacher* and the other of the *student*, which iteratively group images such that their grouping is easier to classify. We describe how we match clusters to the test set in Sec. 3.3.

Formally, let us assume our data contain D domains, with D being a hyperparameter. We initialize two domain

classifiers, the teacher g_ϕ and the student g_φ parametrized by ϕ and φ respectively. Each domain-classifier is a function, mapping images to a probability vector $\mathcal{D}$ defined over the D domains, *i.e.* $g_\cdot : \mathcal{X} \to \mathcal{D}$. Given an input image, the teacher provides domain pseudo-labels as a target to refine student's predictions. In particular, we learn the client student parameters φ_k by iteratively minimizing the cross-entropy loss between the teacher and student domain predictions over $\mathcal{T}_k$. Thus, for a client $k \in K$, the parameters φ_k of the student are:

$$\varphi_k = \arg\min_{\varphi} -\frac{1}{n_k} \sum_{(x,y)\in\mathcal{T}_k} \log g_\varphi^{\hat{d}}(x)\,, \tag{3}$$

where $\hat{d}$ is the pseudo-label given by the teacher for x, *i.e.* $\hat{d} = \arg\max_{d\in D} g_\phi^d(x)$ and $g_*^d(x)$ denotes the probability of x to belong to the d-th domain as given by g_*. Eq. (3) rewards the student from being able to classify according to the pseudo labels, and implicitly encourages agreement on the pseudo-labels, thus on the clustering, which most easily may be agreed upon. Then the domain classifier parameters φ are updated after each round with standard FedAvg, *i.e.* $\varphi = \frac{1}{\sum_{i=1}^{K} n_k} \sum_{k\in K} n_k \varphi_k$.

The idea behind this approach is inspired from deep clustering with self-labelling [45], i.e. the teacher and the student networks would find the equilibrium once they group images in such a way so they can be more easily recognized. This reconnects to the DNNs being natural deep image priors[42], working well for image-related tasks even if just randomly initialized. And it may intuitively match that a "bad" labelling would leave no alternative to a DNN but to overfit [46], which may be hard to imitate by the student. Differently from [45], since we have no access to data and cluster labels, we use the teacher g_ϕ to provide them locally in each client. Both ϕ and φ are randomly initialized and ϕ is fixed during training. After T rounds, with T being a hyperparameter, the parameters ϕ of the teacher are updated with the current student ones φ, iteratively.

Note that, unlike previous works [44], our clustering algorithm can assign unseen data to clusters at test time, thanks to the domain classifier. In particular, the cluster assignment of a test image x corresponds to the domain probabilities given by the student g_φ. Since $g_\varphi(x)$ is soft, we can accommodate for data belonging to unseen domains by a combination of existing ones. Additionally, in our formulation, one client's data samples may belong to multiple clusters, considering the more general case where each client may contain more than one data distribution.

3.2. Cluster-specific Models

Since our model can identify data clusters through the previously described procedure, we can design a way to specialize the function f_θ to each domain. Inspired by multi-domain learning [36, 39, 37, 27, 29], we can achieve this with domain-specific components. For simplicity, let us consider the parameters θ to be split into two sets, *i.e.* $\theta = \{\theta_a, \theta_s\}$ where θ_a are the domain-agnostic parameters and θ_s the domain-specific ones. Note that θ_s is actually a set $\theta_s = \{\theta_s^d\}_{d=1}^D$ where θ_s^d are the parameters specific to the d-th domain. To tailor the model to a specific domain, we can consider multiple ways to include θ_s, such as direct influence on the agnostic parameters θ_a [39, 27, 29] or residual activations [36, 37]. Here we follow the latter strategy, since the former relies on the robustness of θ_a, which is harder to guarantee in FL. Let us assume f_θ to be a deep neural network with a set of layers L, denoting as f_θ^ℓ the function applied at layer $\ell \in L$. Given input from a domain and the features z^ℓ extracted at the previous layer, the output of the ℓ-th layer is:

$$z^\ell = f_{\theta_a}^\ell(z) + \lambda_l \sum_{d=1}^{D} w_d \cdot f_{\theta_s^d}^\ell(z)\,, \tag{4}$$

where λ_l is a learnable parameter balancing the effect of the domain-specific components and w_d is the weight of domain d. During training, we assume data to belong to a single cluster, given by the pseudo-labels of the teacher, thus w_d is 1 if $d = \hat{d}$ and 0 otherwise. At test time, we want our model to deal with data from arbitrary domains by simply combining residuals of seen ones. Thus we set $w_d = g_\varphi^d(x)$, weighting the impact of each domain-specific component by the student output probabilities. Note that the formulation in Eq. (4) is general, with f_θ^ℓ being any layer of a standard convolutional neural network. We explored its application to either the whole network or the last layers.

Since we are in a federated scenario, also the central domain-specific parameters must be updated without access to local data and after each round. In practice, we follow Eq. (2) and we perform FedAvg on both domain-agnostic and domain-specific parameters in each training round.

3.3. Connecting Cluster-specific Models

We now have a model that can adapt to the specificity of each domain. Here we propose to refine the domain-specific parameters by making them interact. Specifically, we model the interaction of the domain-specific parameters of each layer ℓ via a graph $\mathcal{G}^\ell = (\mathcal{V}^\ell, \mathcal{E}^\ell)$, where the nodes $i \in \mathcal{V}^\ell$ are the set of all domain-specific parameters at layer ℓ, and $e_{ij} \in \mathcal{E}^\ell$ are the edges connecting two domain nodes i and j which may interact together. This addresses the drawback of our formulation in Sec. 3.2, *i.e.* if a domain has few assigned samples, its parameters will be rarely updated and thus not robust enough to capture the specificity of the domain and generalize to unseen samples of the same domain.

We propose to use a GCN [18] to model the interaction of domain-specific parameters. Let us collect in the matrix $\mathbf{V}^\ell$

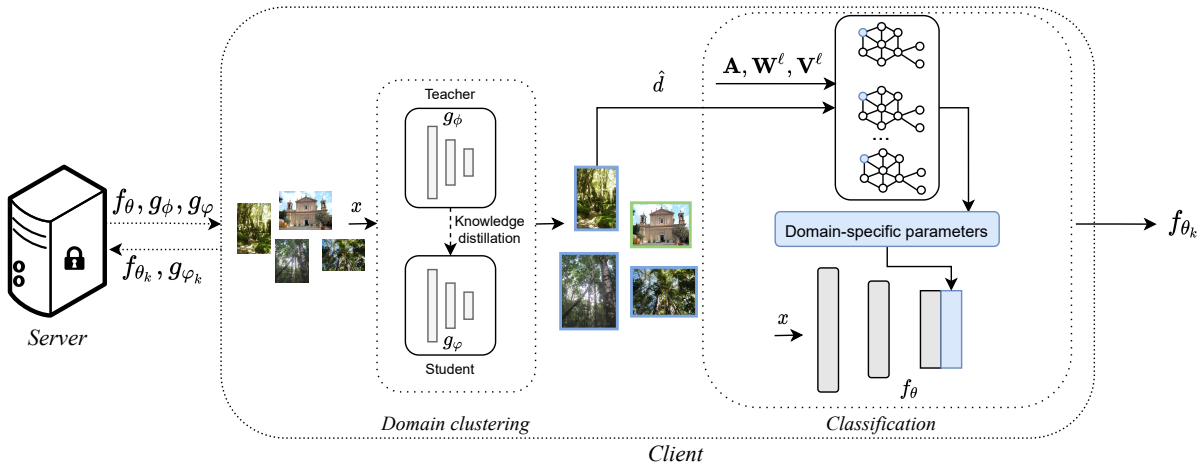

Figure 2. FedCG framework (best seen in colors). The server sends the model f_θ to the clients selected for the federated round, together with the teacher g_ϕ and student g_φ domain classifiers. On the client-side, the domain classifier clusters the local data x, producing as output the domain of belonging $\hat{d}$ of each image. At training time, the hard label $\hat{d}$ is predicted by g_ϕ and is used as input to train g_φ through a process based on knowledge distillation. At test time, $\hat{d}$ is given by g_φ and is a weighted combination of the discovered domains. In FedCG, the network f_θ is made of a domain-agnostic part (in gray) and a residual domain-specific one (in blue). The domain-specific parameters are produced by the GCN, receiving as input $\mathbf{A}, \mathbf{W}^\ell, \mathbf{V}^\ell$ and $\hat{d}$. After training both f_θ and g_φ on its data, the client k sends back to the server the updated weights θ_k and φ_k. On the server-side, the updates are aggregated by the means of the FedAvg algorithm.

the value of each node, *i.e.* all domain-specific parameters at layer ℓ: $\mathbf{V}^\ell = [\theta^1_{s|l}, \dots, \theta^D_{s|l}]^\intercal \in \mathbb{R}^{D \times q}$, with $q = |\theta^d_{s|l}|$ the number of parameters per domain. We compute the graph-version $\hat{\mathbf{V}}^\ell$ of the domain-specific parameters $\mathbf{V}^\ell$ as:

$$\hat{\mathbf{V}}^\ell = \sigma(\mathbf{A}\ \mathbf{V}^\ell\ \mathbf{W}^\ell)\,, \tag{5}$$

where σ is an activation function (*e.g.* ReLU), $\mathbf{A} \in \mathbb{R}^{D \times D}$ is the adjacency matrix defined across the domains, and $\mathbf{W}^\ell \in \mathbb{R}^{q \times q'}$ is a weight projection matrix, projecting the domain-specific parameters into dimension q'. Here, for simplicity, we set $q = q'$. In FedCG, we replace the domain-specific parameters of Eq. (4) with the ones computed in Eq. (5). Similarly to all other parameters of the network, we update $\mathbf{W}$ in each training round through FedAvg. In case q is large, we implement $\mathbf{W}$ as a multi-layer bottleneck (see implementation details).

The values in the adjacency matrix encode, for each edge, how close two domains are; since we have no priors on the structure of the graph, we model $\mathcal{G}^\ell$ as a fully-connected weighted graph. Without direct access to the data server-side, we compute the distance among two domains directly in the (domain-specific) parameter's space. In practice, we define the similarity $h_{i,j}$ among domains i, j as:

$$h_{i,j} = \frac{1}{\|\theta^i_s - \theta^j_s\|_2}\,, \tag{6}$$

and the corresponding value $\mathbf{A}_{ij}$ in the adjacency matrix as:

$$\mathbf{A}_{ij} = \begin{cases} \beta & \text{if } i = j \\ \frac{(1-\beta)\cdot h_{ij}}{\sum_{d=1}^{D} \mathbb{1}_{i \neq d} h_{id}} & \text{otherwise} \end{cases}$$

where β is a hyperparameter weighing the impact of the self-connection, which we set to 0.5, and $\mathbb{1}_{i \neq m}$ is an indicator function being 1 when $i \neq m$ and 0 otherwise.

In our formulation, each client receives not only the set of parameters θ, but also the adjacency matrix. With this definition, we are forcing the gradient of a domain-specific component to flow to all others through the GCN. Consequently, an update on a domain-specific component will influence all domain-specific parameters, even the ones of the domains not present in the current training round. Moreover, given two domains i, j with $i \neq j$, the influence of j on i in each layer is directly proportional to the adjacency matrix value $\mathbf{A}_{ij}$. This means that the more two sets of domain-specific parameters are close, the higher is their mutual influence. Finally, while the GCN is a way to ensure information flow across domains during training, at inference we can just precompute $\hat{\mathbf{V}}^\ell$ for each layer, to save memory usage.

4. Experiments

4.1. Datasets and implementation details

We evaluate the proposed model on image classification tasks on the LEAF benchmark [2], testing both on the CelebA [25] and Federated Extended MNIST (FEMNIST) [20, 6] datasets. Table 1 details each setting.

CelebA is a widely used dataset containing pictures of faces of several celebrities. We follow the same experimental protocol of [2], partitioning the dataset by celebrities and ignoring the ones with less than 5 images. The task is

binary classification, recognizing whether the depicted person is smiling or not. Following [2], we use 10% of the total clients for training and a separate split of 20% of them for test. We train FedAvg and our model on 100 rounds with 10 clients each, training locally for a single epoch with a batch size of 5 and a learning rate of 10^{-3}. To perform a fair comparison we used the same architecture of [2], replacing convolutional and batch-normalization layers with their FedCG counterpart, based on a 1-layer GCN.

FEMNIST contains images depicting different characters drawn by different writers. The task is a 62-way classification problem, where the classes correspond to the uppercase and lowercase letters of the alphabet and numbers. Following the setting proposed by [2], each client corresponds to a different writer, using 60% of them for training, 20% for validation and 20% for test. We run both FedAvg and FedCG for 1000 rounds of 5 clients each, using a batch size of 10, a learning rate of 10^{-3} and one local epoch. We use the same and architecture of [2], replacing the last convolutional layer with our GCN-based version. In this case, we use a 2-layers GCN, modeling the projection matrix $\mathbf{W}$ as a bottleneck dividing the features by a factor of 16.

Implementation details In all datasets, the domain classifiers are CNNs made of two convolutional layers of 32 and 64 features and kernel size 3×3, followed by an average pooling and a linear layer whose output dimension is the number of domains. We train the domain classifiers through an SGD optimizer without weight decay and a learning rate of 10^{-4}. We implement FedCG on PyTorch [35], running the experiments on NVIDIA GeForce 1070 GTX GPUs. We chose PytTorch due to its higher flexibility for prototyping and experimenting the components of our model. To ensure a fair comparison, we implemented the FedAvg baseline using the same framework, architectures, hyperparameters and training protocols of [3].Table 2 compares our results with the performance of the Tensorflow [1] implementation from the LEAF repository [3]: our FedAvg baseline outperforms the original one by almost 3% in accuracy on FEMNIST, while performing almost 2.5% less on CelebA. Nevertheless, our main interest is to evaluate the relative improvement of the proposed model with respect to a baseline that does not exploit domain information, using the same aggregation strategy of FedAvg for federated learning. For these reasons, in the following we will take as reference the FedAvg results of the PyTorch framework, to ensure a comparison with the baseline under the exact conditions. We evaluate our results in terms of global accuracy on the test set, *i.e.* on the union of the images of all test devices. All experiments on the same dataset were run with the same configuration to perform a fair comparison between the considered approaches.

Dataset	Clients	Total samples	Samples per client		Classes
			Mean	Stdev	
CelebA	9,343	200,288	21.44	7.63	2
FEMNIST	3,550	805,263	226.83	88.94	62

Table 1. **Datasets Statistics**

Dataset	TensorFlow	PyTorch
CelebA	89.46	86.88
FEMNIST	74.72	77.81

Table 2. Accuracy of FedAvg in TensorFlow [2] and our version implemented in PyTorch.

4.2. Ablation study

In this section, we focus our analysis on testing the performance of the proposed model on the *CelebA* dataset, analyzing the various components of our approach. All referred studies and results can be found in Table 3 and 4.

4.2.1 How to use domain information

To create a sanity-check for our model, we first define the domains manually, exploiting the a priori knowledge given from the images meta-data, *i.e.* the 40 attributes of the dataset (Table 3). This allows us to isolate the choice of how to include domain-specific information within the model, without any influence from the clustering procedure. From the 40 attributes, we select the combination of n attributes leading to the most balanced subdivisions of the dataset and having a low correlation with the target feature. Since each attribute can only assume the values $\{0, 1\}$, the number of possible *domains* is given by all the 2^n combinations of the n features. We choose $n = 5$, having $N = 32$ domains. The selected features are *attractive*, *heavy makeup*, *high cheekbones*, *mouth slightly open* and *wavy hair*.

Domain-specific models We start by replacing the standard single server model with N separate domain-specific models, trained and tested only on the images of their specific domains. As shown in Table 3, the performance drops significantly (33.61% vs 86.88% of FedAvg), since the insufficient amount of data seen by each model leads to poor generalization. This shows that learning a single full model per each domain is not a viable strategy in this scenario.

Modeling the relations across domains In order to account for the relations existing among the different domains, we introduce the graph, modeled as a 1-layer GCN. To study the impact of introducing a GCN, we also test a simpler version of the model without the weight transformation matrix W. We analyze different choices of $\mathbf{A}$, considering the cases where i) they are uniformly weighted (U) and ii) the domains are weighted according to a similarity criterion (H), specifically the normalized inverse of the Hamming distance [34] between the numerical representation of the domains (*i.e.* their binary metadata). As Table 3 shows, using a GCN consistently improves the final performance over the domain-specific models. The uniform adja-

cency matrix performs slightly better than the weighted one in this case, with both their performance improving when the projection matrix $\mathbf{W}$ is introduced. These results confirm the importance of making the domain-specific nodes interact. However, the results are not satisfactory, being either below or just 1% above (GCN-H with $\mathbf{W}$) FedAvg. This means that domain information is still dully exploited within the model.

Residual domain-specific layers. Finally, we analyze the usage of domain-specific parameters to produce residual activations (*i.e*. Eq. (4)), as in FedCG, comparing it with the GCN when not using any domain-agnostic component. As Table 3 shows, while the model with uniform adjacency matrix (U) sees a decrease in performance from GCN to FedCG (*i.e*. 87.92% vs 86.96%), the model with weighted adjacency matrix (H) sees a large boost, going from the 84.25% accuracy of GCN to the 88.65% of FedCG. We can draw two conclusions. First, using residual layers to refine the domain agnostic activations (FedCG) performs better than using only domain-specific components (GCN). Second, when domain-specific components are integrated as residuals, they are much more effective when connected in a weighted (H) rather than a uniform (U) fashion. This is proved by the results of FedCG-H, surpassing FedCG-U by 1.7% in accuracy. Finally, we test the importance of the ReLU non-linearity applied to the output of the residual GCN. The non-linearity improves FedCG, both when the domains are connected uniformly (+1%) and in a weighted fashion (+0.9%). The final FedCG model with 1-layer GCN filtered with a ReLU and a weighted adjacency matrix outperforms the baseline FedAvg by 2.6% accuracy, showing the effectiveness of our choices.

4.2.2 How to identify the domains

In the previous section, we analyzed how to integrate domain-specific components given oracle domain information. In this section, we drop the assumption of having such information and we study the effectiveness of the domains discovered through our clustering procedure on FedCG held out through the teacher-student domain classifier (cf. Sec. 3.1). We report the results of our analysis in Table 4.

Domains extracted through clustering We start by comparing our domain classifier with the K-means algorithm [26] applied to the parameters of the models trained separately on each client. FedCG performs clustering locally instead, accessing only a subset of the clients at each round. As Table 4 shows, the performance of our clustering procedure is either on par ($D = 2$) or superior ($D = 3,4$) to K-means clustering. In particular, as the number of clusters grows, the performance of K-means drops (*i.e*. from 88.36% with $D = 2$ to 87.21 with $D = 4$), while our method - with the same residual GCN - shows performance improvements (*i.e*. from 88.03% with $D = 2$ to 88.74 with $D = 4$). This indicates the effectiveness of our local clustering procedure that, differently from K-means, captures the presence of different domains within each client, without requiring one specific model per client.

Then, we analyze the effect of different initialization strategies for the adjacency matrix of the GCN, considering two choices, *i.e*. domains either disconnected (identity matrix, *eye*) or randomly connected (random adjacency matrix, *rand*). From Table 4, it is easy to see our method performances are not dependent on the particular initialization strategy, achieving over 88.5% for all choices with $D = 4$. With random initialization though, the performance does not grow with the number of domains, which may indicate the importance of carefully initializing $\mathbf{A}$ as the number of domains grows. For this reason, in the following we always consider a uniform initialization strategy. Note that such a strategy allows the model to refine the domain-specific components separately before merging them based on their distance (see Eq. (3.3)).

As a third analysis, we focus on the impact of performing a soft combination of domains at test time (as described in Section 3.2) rather than using a hard-assignment derived from the predictions of the domain classifier. In both cases, performances are close for all D, showing the domain classifier provides reliable domain predictions at test time. In the following, we always consider the soft-assignment due to its higher flexibility.

Finally, we test the application of the domain-specific modules only on the last layer of the network (rather than on all layers) to see whether a good performance can be achieved while reducing the number of parameters required by FedCG. As the experiments show, using domain-specific parameters on the last layer provides the best results overall (89.18% with $D = 4$), improving the best combination by 0.44%. Since this choice allows FedCG to use less parameters while still achieving good results, we limit the use of domain-specific parameters to the last layers in the next section.

4.3. Comparison with the state of the art

Here we compare our FedCG with state-of-the-art results on both CelebA and FEMNIST. Unfortunately, since different methods employ different settings and client splits, it is difficult to provide an extensive comparison on these datasets. For this reason, on CelebA we compare our model directly with the FedAvg baseline, while for FEMNIST we compare it with FedAvg, FedProx [23] and SCAFFOLD [16]. FedProx [23] adds a proximal term to the standard FedAvg algorithm for improving the model stability when applied over heterogeneous systems and data. SCAFFOLD [16] uses variance reduction for minimizing the impact of the drift in the updates of each client. We report the results of FedProx and SCAFFOLD shown in their original papers,

Model	A	W	ReLU	Acc(%)
Domain-specific models	-	-	-	33.61
GCN	U	✗	✗	84.39
	H	✗	✗	82.10
	U	✓	✗	87.92
	H	✓	✗	84.25
FedCG	U	✓	✗	86.96
	H	✓	✗	88.65
	U	✓	✓	87.97
	H	✓	✓	89.57

Table 3. **Ablation studies on CelebA dataset with $N = 32$ domains extracted from images meta-data**. **A** is the adjacency matrix that weights the domains contributions: the symbols (eye,U,H) respectively stand for identity, uniform and weighted (with inverse Hamming distance) matrices. **W** is the weight projection matrix and ReLU the chosen non-linear activation.

FedCG layers	**A** init	Clusters	D	Soft domains	Acc(%)
all	eye	K-means	2	✗	88.36
	eye	K-means	3	✗	87.97
	eye	K-means	4	✗	87.21
	eye	Clf	2	✗	88.03
	eye	Clf	3	✗	88.59
	eye	Clf	4	✗	88.74
	rand	Clf	2	✗	88.73
	rand	Clf	3	✗	88.24
	rand	Clf	4	✗	88.55
	eye	Clf	2	✓	87.88
	eye	Clf	3	✓	88.74
	eye	Clf	4	✓	88.67
last	eye	Clf	2	✓	88.31
	eye	Clf	3	✓	88.13
	eye	Clf	4	✓	**89.18**
	eye	Clf	32	✓	88.40

Table 4. **Ablation studies on CelebA dataset with domains given by a priori knowledge or online clustering procedures**. In the **A** init column, "eye" stands for identity matrix and "rand" for random. The third column specifies the clustering, *i.e.* clusters generated with K-means or the teacher-student classifier ("Clf").

while for FedAvg we use our baseline. For our method, we use the domain-specific parameters applied on the last layer, $D = 4$, soft domain assignments at test time and the adjacency matrix initialized as identity.

The experimental comparison is reported in Table 5. FedCG largely outperforms FedAvg in both scenarios. It achieves 89.18% accuracy compared to 86.88% of FedAvg on CelebA, and 83.41% accuracy compared to 77.81% of FedAvg on FEMNIST. This latter improvement (+5.6%) is remarkable given the higher complexity of the classification task in FEMNIST. Comparing FedCG with FedProx and SCAFFOLD on FEMNIST, we can see that FedCG outperforms FedProx by a large margin (+8.41%) while being slightly inferior to SCAFFOLD (*i.e.*-0.79%). However, both FedProx and SCAFFOLD present results under different federated protocols, *e.g.* FedProx runs the algorithm for

Dataset	**Model**	**Accuracy (%)**
CelebA	FedAvg	86.88
	FedCG	**89.18**
FEMNIST	FedAvg	77.81
	FedCG	**83.41**
	FedProx	75.00
	SCAFFOLD	84.20

Table 5. **Comparison with the state of the art on CelebA and FEMNIST.** We separate the methods according to their setting.

200 rounds of 10 clients while SCAFFOLD performs 1000 rounds with 20 clients each. Despite that, our comparisons demonstrate that FedCG is far superior to the standard FedAvg baselines, due to its better ability to address the statistical heterogeneity across clients, while showing either superior (w.r.t. FedProx) or competitive (w.r.t. SCAFFOLD) results with other state-of-the-art algorithms trained on different settings.

As a final analysis, we verify the role of the domain-specific components by checking the final values of the λ scalar of Eq. (4), which weighs the importance of the domain-specific residual. Interestingly, in CelebA, where the concept of heterogeneity is less marked, the λ value in the last convolutional layer is 0.3. The final value for FEMNIST, instead, where the heterogeneity across clients is clearer due to the different writing styles, is 2.1. That shows FedCG tailors the use of the domain-specific residual to the specific characteristics of the target dataset and the consequent heterogeneity across the discovered domains.

5. Conclusions

In this work, we introduced FedCG, the first cluster-driven approach addressing statistical heterogeneity in federated learning with Graph Convolutional Neural Networks. FedCG uses an iterative clustering algorithm based on teacher and student domain classifiers. This clustering procedure serves to discover different input distributions, *i.e.* domains, and to instantiate domain-specific parameters accordingly. The domain-specific parameters are connected through a GCN that enables them to interact and share knowledge during training. These parameters influence the activation of the main, domain-agnostic, network thanks to weighted residual activations. Thanks to the domain classifiers and connections of the GCN, new input distributions and unseen users can be addressed at test time via their domain soft-assignment scores. Experimental results show that FedCG outperforms the FedAvg on multiple benchmarks, demonstrating the efficacy of each component.

Acknowledgments This work has been partially funded by the ERC 853489 - DEXIM, the ERC 802554 - SPECGEO, the ERC 637076 RoboExNovo, the DFG – EXC number 2064/1 – Project number 390727645, and the MIUR under grant "Dipartimenti di eccellenza 2018-2022".

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.

[2] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[3] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. https://github.com/TalwalkarLab/leaf, 2018.

[4] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[5] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[6] 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.

[7] Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[8] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3557–3568. Curran Associates, Inc., 2020.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[11] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification, 2019.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 76–92, Cham, 2020. Springer International Publishing.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[14] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[15] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

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

[17] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[18] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[19] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[20] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[21] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. In *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[22] 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.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[24] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231, 2019.

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

[26] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[27] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[28] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Adagraph: Unifying predictive and continuous domain adaptation through graphs. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 6568–6577, 2019.

[29] Massimiliano Mancini, Elisa Ricci, Barbara Caputo, and Samuel Rota Bulo. Boosting binary masks for multi-domain learning through affine transformations. *Machine Vision and Applications*, 31(6):1–14, 2020.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[31] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. Sgnn: A graph neural network based federated learning approach by hiding structure. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2560–2568. IEEE, 2019.

[32] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4615–4625. PMLR, 09–15 Jun 2019.

[33] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[34] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 506–516, 2017.

[37] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[38] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020.

[39] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

[40] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[41] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

[43] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. Graphfl: A federated learning framework for semi-supervised node classification on graphs, 2020.

[44] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning. *arXiv preprint arXiv:2005.01026*, 2020.

[45] Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020.

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[47] Longfei Zheng, Jun Zhou, Chaochao Chen, Bingzhe Wu, Li Wang, and Benyu Zhang. Asfgnn: Automated separated-federated graph neural network. *Peer-to-Peer Networking and Applications*, pages 1–13, 2021.