Many-Task Federated Learning: A New Problem Setting and A Simple Baseline

Ruisi Cai[1], Xiaohan Chen[2], Shiwei Liu[1],
Jayanth Srinivasa[3], Myungjin Lee[3], Ramana Kompella[3], Zhangyang Wang[1]
[1]University of Texas at Austin, [2]Alibaba US, [3]Cisco Systems
{ruisi.cai, atlaswang}@utexas.edu xiaohan.chen@alibaba-inc.com shiwei.liu@austin.utexas.edu
{jasriniv, myungjle, rkompell}@cisco.com

Abstract

Federated Learning (FL) was originally proposed to effectively exploit more data that are distributed at local clients even though the local data follow non-i.i.d. distributions. The fundamental intuition is that, the more data we can use the better model we are likely to obtain in spite of the increased difficulty of learning due to the non-i.i.d. data distribution, or data heterogeneity. With this intuition, we strive to further scale up FL to cover more clients to participate and increase the effective coverage of more user data, by enabling FL to handle collaboration between clients that perform different yet related task types, i.e., enabling a new level of heterogeneity: ***task heterogeneity****, which can be entangled with data heterogeneity and lead to more intractable clients. However, solving such compound heterogeneities from both data and task levels raises major challenges, against the current global, static, and identical federated aggregation ways across clients. To tackle this new and challenging FL setting, we propose an intuitive clustering-based training baseline to tackle the significant data and task heterogeneities. Specifically, each agent dynamically infers its "proximity" with others by comparing their layer-wise weight updates sent to the server, and then flexibly determines how to aggregate weights with selected similar clients. We construct new testbeds to examine our novel problem setting and algorithm on two benchmark datasets in multi-task learning: NYU Depth and PASCAL-Context datasets. Extensive experiments demonstrate that our proposed method shows superiority over plain FL algorithms such as FedAvg and FedProx in the 5-task setting on Pascal-Context and even enables jointly federated learning over the* ***combined set*** *of PASCAL-Context and NYU Depth (****9 tasks, 2 data domains****). Codes are available at: https://github.com/VITA-Group/MaT-FL.*

1. Introduction

Deep neural network training can significantly benefit from a substantial amount of data, which is largely created by edge devices. To make full use of these data while preserving data privacy, the federated learning (FL) [8, 14] is introduced to enable many clients with decentralized data to collaboratively train a model under the coordination of a central server. In the traditional FL setting, due to the various deployment environments and clients' characteristics, participating clients typically tackle highly diverse samples (i.e. data heterogeneity). e.g., two vehicles performing lane detection in urban and rural areas respectively would gather dissimilar image samples of lanes from two different distributions. Such non-IID sample distribution among clients is detrimental to the optimization [10, 11, 27], and becomes one of the key challenges of FL. While there are challenges, the fundamental intuition behind the introduction and the success of FL is that, compared to learn models at local clients with an extremely limited amount of samples, the more data we can use the better model we are likely to obtain in spite of the increased difficulty of learning.

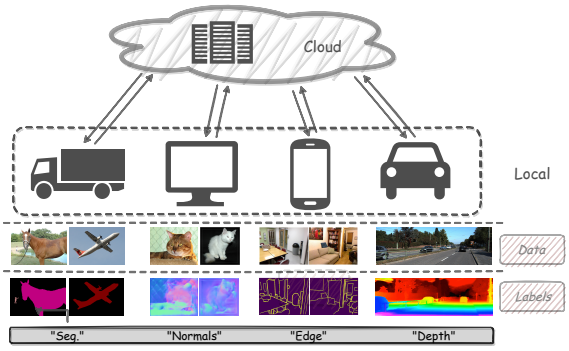

Figure 1. The illustration of the proposed many-task federated learning (MaT-FL) setting. Each client is dedicated to its own task and has only access to the labels of that task. The goal of this work is to establish an FL framework that allows all clients that perform different tasks (e.g. semantic segmentation (Seg.), surface normals estimation (Normals.), edge estimation (Edge), depth estimation (Depth) to collaboratively train a model together[1].

[1]The work is sponsored by a Cisco Research Grant (FA00000274).

1.1. Many-Task Federated Learning: Problem Setting, Challenges & Opportunities

Due to the increasing demand for one real-world ML system to handle different tasks, multi-task learning (MTL) is emerging [5, 18, 22]. Tempted by the intuition of including more data from edge clients in training, MTL model trainers are also compelled to turn to FL approaches, which bring about not only extra data but also more intractable clients. Specifically, besides samples having different data distributions among clients, the assumption that every client tackles the same set of tasks which naturally holds in the centralized learning case cannot be expected anymore. The rationality lies in: (1) The process of obtaining task labels often requires tedious and expensive labor; hence not all clients can have access to all task labels. (2) In many real-world scenarios, each client would only specialize in one task. In such scenarios, each client may train a local model dedicated to its own task which differs from other clients in the same network. We term this as "task heterogeneity", which is a new and under-explored form of heterogeneity in FL that makes the learning process even more challenging.

Note that the new task heterogeneity could co-exist with and reinforce the existing data heterogeneity since task-diverse clients might tend to acquire their local data from diverse distributions too. However, besides the amplified challenges, we argue that FL and MTL also have complementary merits for their co-design:

- MTL can potentially improve FL: (1) MTL relaxes the restriction on clients, enabling FL to unify more diverse data for training. This is well aligned with the intuition of FL to exploit as much data as possible and is also consistent with our empirical observations later in this work. (2) if FL systems can learn a common model for many tasks, then it could lead to more robust and universal feature representation [26] that can scale to more clients, data & task heterogeneity.

- FL can in turn provide a natural solution for the inherent MTL issue of cross-task conflicts. Prior works [2, 24] reveal that naively training all tasks together in one model often lead to degraded shared representations, due to different tasks often incur different update directions that might comprise each other. While existing proposals [2, 15, 24] still train one model backbone with shared weight, the distributed and "modularized" learning nature of FL agents could be leveraged as a blessing to disentangle the parameter spaces, enabling MTL model trainers flexibly group and aggregate models as needed.

1.2. Our Contributions

This paper aims to establish an algorithm framework that enables such "many-task federated learning" (MaT-FL), i.e., clients specializing on different tasks yet still collaborating through FL. As a preliminary baseline, we introduce dynamic grouping as our core countermeasure against strong data and task heterogeneity: (1) following the practice of personalized FL [7, 25], each FL client will perform its own weight aggregation (rather than aggregating to one global model), and (2) each client will adaptively identify its "friendly" clients while excluding those whose local updates severely conflict its own. Using this flexible yet heterogeneity-aware mechanism, we aim to strike the balance between utilizing more diverse data and avoiding training conflicts. To evaluate the effectiveness of the proposed method, we also construct new FL testbeds from two multi-task learning benchmarks: NYU Depth [17] and PASCAL-Context [16]. In their FL settings, while all clients jointly learn multiple tasks, each client has access to a subset of data and label belonging to only one task.

Our contributions in this work are summarized below:

- We introduce the "many-task federated learning" (MaT-FL) setting where local clients can collaborate effectively even they are specialized at different tasks. This promising new setting could lead to scaling up FL "in the wild" by covering more clients, and hence to much broader data coverage and effective utilization.

- We leverage the dynamic grouping mechanism for each agent to dynamically choose and aggregate weights from its friendly neighbors (by comparing their update direction similarity). That effectively tackles the gradient conflicts during federated aggregation caused by task heterogeneity.

- Extensive experiments demonstrate that our method outperforms off-the-shelf alternatives such as FedAvg and FedProx in both new benchmarks for MaT-FL. Most impressively, our algorithm even enables jointly FL over the **combined set** of Pascal-Context and NYU Depth (**9 tasks, 2 data domains**), setting a new record of FL under strong data and task heterogeneity.

2. Related Work

2.1. Federated Learning

Federated learning (FL) [8, 10, 11, 14, 27] allows local clients to collaboratively train models without sacrificing data privacy. Normally, it involves two stages: (i) local clients train models separately and send models to the cloud, and (2) the cloud aggregate those models to obtain a global model which will be send back to local clients for next round of training. Several methods concentrate on handling heterogeneity of local data distribution across clients. For instance, FedProx [11] improves FedAvg [14] by adding a proximal term into the local training objective

function to limit the norm of local updates; FedNova [23] instead aggregate models with consideration in the different computational power of clients; SCAFFOLD [9] corrects local updates through local control variates.

2.2. Personalized Federated learning

Instead of assuming a single global model to accommodate multiple domains, personalized FL methods [7, 25] aims to tackle heterogeneity through learning a personalized global model for each individual client, enabling stronger compatibility with highly non-IID clients. These methods typically empower each client to play a more active role in deciding who to collaborate with and how to aggregate their models. [25] allows each client to assess the degree to which other models can facilitate local convergence on a hold-out validation set. However, this approach requires intensive computational and communication costs and hence may not be scalable to scenarios with many participating clients and under resource constraints. [7] encourages collaboration between similar clients through an attentive message-passing mechanism that learns the relations of clients implicitly. [4] introduces a clustering algorithm to depict relationships of clients, where aggregations are only computed within the group of clients. This approach assumes mutual relationships between the models, whereby if one model helps another, the other model will reciprocate and provide help as well, which may be invalid in practical MTL settings where tasks may either help or inhibit each other. Compared to those methods, our algorithm intends to introduce a simple, straightforward baseline and handles heterogeneous task types (i.e., segmentation, depth, normal...; rather than single task type such as classification, just with different per-client classes) - a more challenging case that none of the prior works have accounted for.

2.3. Multi-Task Learning

Multi-task learning (MTL) [5, 12, 18, 22] aims to leverage useful knowledge contained in multiple tasks to improve the generalization of features and also suits the need of handling different tasks in the same model. As task conflicts being the major problem in MTL [2], several works explore grouping-based methods. For example, [15] introduce "cross-stitch" units; [2] apply multi-task loss balancing; and [24] use gradient projection to deal with gradients with negative cosine similarity; and TAG [3] exploited task affinities (from a pre-trained MTL model) for task grouping. However, their methods are all "centralized", i.e., assuming the model trainer has access to all tasks and data.

2.4. Federated Learning with Multi-Task Objective

There exist prior works attempting to combine MTL and FL. [20] addressed the MTL-based FL setting for the first time, learning separate models for each client. However, their discussion was primarily based on learning linear convex models, and cannot be straightforwardly generalized to non-convex deep learning models. [13] and [19] tackle data heterogeneity problems but did not assume local clients to handle heterogeneous task types. The recent work [28] is most similar to ours, but with totally different motivations and setting. They assume each client to have a copy of the full MTL model and a subset of data from *all tasks*, with their goal to have clients jointly train the MTL model using FL. Their work hence was not motivated by nor contributing to broadening the data coverage (from diverse single-task clients). In contrast, we make the more realistic assumption that each client only accesses a data subset from one *single task*; and each client trains its own single task model while only the cloud aggregates multiple client/task information.

3. Methodology

3.1. Problem Formulation

For an MaT-FL system with N clients and C tasks, client i owns data $D_i = \{X_i, Y_i^t\}, i = 1, 2, ..., N$, where Y_i^t denotes the label of sample X_i on task t. Each client i specializes on a specific task $t \in \{t_1, t_2, ..., t_C\}$ whose relationship can be formulated through a mapping between the client index and the type of task $t = \mathcal{T}(i)$. Training on the local data, each client will obtain the local update g_i^t by:

$$g_i^t = \nabla_{\theta_i^t} \mathcal{L}(X_i, Y_i^t) \tag{1}$$

An important change to note is at the cloud side: we assume the MTL model trainer to maintain a **task-specific global model** θ^t for each task t, hence C global models in total, by certain weights aggregation strategy to be detailed next. Those global models θ^t will be correspondingly dispatched to clients performing the same task types in the next communication round.

3.2. Preliminaries

Single-task FL: In traditional FL approach FedAvg [14], without incorporating multi-task scheme, the global model θ^t solely focus on same-task clients and ignore other data.

$$g^t = avg_i\{g_i^t, i : \mathcal{T}(i) = t\} \tag{2}$$

Plain Many-task FL: A natural way to utilize full data is to directly extend FedAvg [14] into multi-task scenario without considering both data and task heterogeneity problems. This way will ignore the difference between clients.

$$g^t = avg_i\{g_i^t, i = 1, 2...N\} \tag{3}$$

Existing advanced FL: With many recent approaches trying to tackle heterogeneity problems, as the extension of FedAvg [14], the representative work FedProx [11] add the

proximal term in the local training to restrict the local updates to be closer to the initial model. However, as MTL bringing "task heterogeneity" of clients, such method is not capable of handle this more severe heterogeneity challenge.

$$g_i^t = \nabla_{\theta_i^t} \mathcal{L}_{\text{prox}}(X_i, Y_i^t)$$
$$g^t = avg_i\{g_i^t, i = 1, 2...N\} \tag{4}$$

3.3. Proof of Concepts: Why Grouping

First, let us find out the characteristic of MaT-FL weight updates. To be specific, we want to zoom in to one single FL step, and answer the following question:

With the same weight starting point, do models performing different tasks update towards the dissimilar directions?

To answer the question, we train 90 single-task models on different datasets and tasks, each for 10 epochs, with the same starting point of ImageNet-pre-trained ResNet18 [6]. For Pascal-Context dataset [16], we train models on: Semantic Segmentation; saliency estimation; Surface Normals Estimation; Edge Estimation; human part segmentation. For NYUD-v2 dataset [17], we train models on: Depth Estimation; Semantic Segmentation; Surface Normals Estimation; Edge Estimation. For each dataset-task combination, we independently train 10 models with random 10% data subsets. We then collect all their weight updates from the starting point, and visualize those weight updates by T-SNE [21] in Figure 2. Note that we normalize the updates first since directions of updates are more likely to lead to conflicts [2, 24]. We draw two observations:

- The updates are well-clustering by tasks, which means models performing the same task would behave substantially more similarly. Such similarity can even be across-dataset: for example, Pascal updates from Semantic Segmentation are even more similar to NYUD updates of the same task type, compared to other tasks on Pascal-Context. That shows compared to data sources (domains), the task differences make the more dominant source of heterogeneity.

- For each task, there are "*friends*" and "*enemies*", in terms of similar/dissimilar normalized weights updates, i.e., different update directions.

The above observations motivate us to design a mechanism to strike a balance between covering more "friends" and preventing training interference caused by "enemies". At the heart of the mechanism is our new regime to effectively group the "friends" and "enemies" for each client, and to adaptively aggregate updates within friend groups.

Figure 2. The T-SNE [21] visualization result for updates using data on different tasks, with the same ImageNet-pre-trained ResNet18 as starting point.

3.4. Dynamic Grouping and Adaptive Aggregation

The pipeline of our method is illustrated in Figure 3 and summarized in Algorithm 1. In each communication round, our method can briefly separated into two parts: for each client, we first group other clients into "friends" and "enemies", then generate aggregated updates within the group.

STEP 1: Dynamic Grouping. To explicitly tackle highly dissimilar clients, we introduce the dynamic grouping technique to let each client decide whom others to aggregate with. Note that considering the training dynamics, we do grouping in each communication round independently instead of grouping once and sticking to it.

Algorithm 1 Pseudo-codes for Dynamic Grouping and Adaptive Aggregation

Input: Local updates collected by cloud g_i^t.
  for i=0 to $N-1$ **do**
    for l=0 to $L-1$ **do**
      $s_{i,j}^{(l)} = \cos\{g_i^{t,(l)}, g_j^{t,(l)}\}$
      $votings_i^{(l)} = \text{TopK}(s_{i,j}^{(l)})$
    end for
    $\mathcal{P}(i) = \text{Avg}\{votings_i^{(l)}\}$
    $\hat{a}_i^t = \text{Aggregate}\{g_j^t, j \in \mathcal{P}(i)\}$
  end for
  for t=0 to $K-1$ **do**
    $\theta^t = \text{Avg}_i\{\hat{a}_i^t, \mathcal{T}(i) = t\}$
  end for
Output: Task-specific global models θ^t.

In each round, after the cloud collecting all local updates

$\{g_i^t, i = 1, 2...N\}$, we first compute the similarity score between each pair of clients for each layer. At layer l, we obtain the score $s_{i,j}^l$ between clients i and j by computing the normalized cosine similarity between their weight updates:

$$s_{i,j}^{(l)} = \cos\{g_i^{t,(l)}, g_j^{t,(l)}\} \tag{5}$$

Each layer will then independently votes. Specifically, in each layer, we will pick K clients with the highest K similarity scores and assign related clients with vote values in descending order (the most similar one assigned with "K", the second most similar "K-1", ...). Next, for each client i, we will collect all its layers' votes and pick the top "friends" $\mathcal{F}(i)$ with the highest accumulated vote from all layers. Aggregation of this client i's update in the next step will only be performed within the group $\{g_i^t, i \in \mathcal{F}(i)\}$.

STEP 2: Adaptive Aggregation. After grouping, we generate aggregated updates $\hat{a}_i^t$ for each client. To further cope with the potentially conflicting update directions even within a group, we leverage similarity scores to adaptively reweight local updates instead of directly averaging.

After obtaining the grouping result $\mathcal{F}(i)$ for each client i, we consider re-using the similarity scores generated by Equation 5 for aggregation. To be specific, we first generate normalized similarity score $\hat{s}_{i,j}^l$ to keep $\sum_j \hat{s}_{i,j}^l = 1$ and average them on each layer to obtain the final similarity score $s_{i,j} = \text{Avg}_l\{\hat{s}_{i,j}^l\}$. Then, for every client j in $\mathcal{F}(i)$, we use $s_{i,j}$ as weights, generating the aggregated update $\hat{a}_i^t$. We perform normalization to have $\hat{s}_{i,j}$sum up to 1:

$$\hat{a}_i^t = \sum_{j \in \mathcal{P}(i)} \{\hat{s}_{i,j} * g_j^t\} \tag{6}$$

Finally, we update the global model θ^t for each task t using the average of $\{\hat{a}_i^t, \mathcal{T}(i) = t\}$. Such models will be dispatched in the next round of federated learning.

4. Experiment Results

4.1. Implementation Details

We leverage two representative MTL datasets, Pascal-Context [16] and NYU Depth v2 (NYUD) [17]. Pascal-Context dataset [16] has annotations on five tasks: semantic segmentation (Seg.); saliency estimation (Sal.); surface normals estimation (Norm.); edge estimation (Edge); human part segmentation (H.Parts). NYUD dataset [17] has labels for four tasks: depth estimation (Depth); semantic segmentation (seg.); surface normals estimation (Norm.); edge estimation tasks (Edge). From those two datasets, We establish two benchmark settings for MaT-FL:

- **Pascal-Context (5 tasks)**: all clients have a subset of Pascal-Context and C equals 5. Each client specializes on one of five tasks.
- **Pascal-Context & NYUD Combined Set (9 tasks)**: some clients have a subset of Pascal-Context while others have a subset of NYUD. We consider C equals 9 in this setting, by treating the overlapped tasks between the two datasets (such as Seg.) as two different tasks. Each client specializes on one of nine tasks.

In either setting, we randomly partition the original dataset evenly for N clients. For simplicity, we set the number of clients accounting for each task to be the same, i.e., N/C. We can control N/C to adjust the scale of FL, and we set it to be 2 by default, while more ablations are in Section 4.3.

We use ResNet18 as the backbone and deeplab [1] as decoder on the head. In all experiments, we set local training epochs as 10 and the number of total communication rounds as 100. We use a batch size of 10 and an initial learning rate of 10^{-4}. After every communication round, we decay the learning rate by a factor of 0.99.

4.2. Improvements via Dynamic Grouping

We conduct experiments on the aforementioned settings and hyper-parameters. For our proposed method with dynamic grouping, we set K in TopK as 8, and we will explore its effect in Section 4.4. We provide results in Table 1. Here, the terms "Single-Task FL", "Plain Many-Task FL", and "FedProx" are illustrated in Section 3.2. We show the evaluation results of two representative tasks and provide others in the Appendix. On both Pascal-Context(5) and Pascal-Context & NYUD(9) scenarios, we have several interesting observations: ❶ Despite the potential interference brought by task heterogeneity, directly extending Single-Task FL into Many-Task FL can help unify more data, benefiting the training, which echoes the "the more data, the better model" intuition in Section 1. ❷ FedProx fails to tackle task heterogeneity and barely brings performance gain. ❸ After using dynamic grouping and adaptive aggregation, models can be free of interference and further benefit from more data. Specifically, on Pascal-Context(5), compared to Single-Task FL, our method achieves improvements of 0.0313, 0.0155 on Seg. Sal., respectively. Similarly, on Pascal-Context & NYUD(9), performances increase by 0.0489, 0.0195.

4.3. Scaling the number of clients per task

To further validate our effectiveness, we test our method on FL systems of different scales. To be specific, on the Pascal-Context(5) setting, we change the number of clients per task to control how many clients are involved in training. According to Table 2, under different FL scales, our method consistently shows better performance than baseline methods and similar patterns with results in Section 4.2. Surprisingly, our performance is even better than baseline methods with the smaller FL scale, which validates the effectiveness

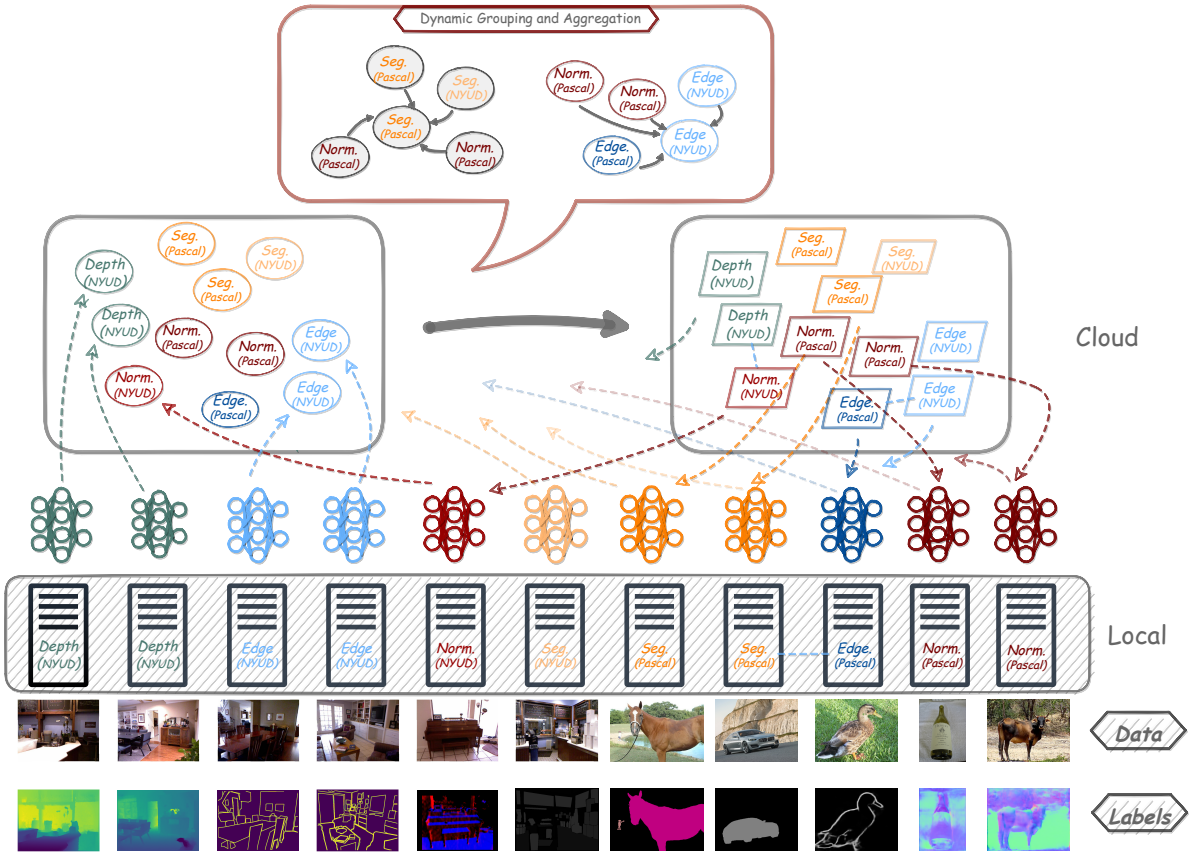

Figure 3. Illustration of how we could scale up FL by explicitly handling task heterogeneity. Our algorithmic framework is to dynamically identify and group similar "friends" for each client during training, and then adaptively aggregate weights within the same group. Note that ovals represent updates collected from edge devices, while parallelograms represent updates after aggregation.

Table 1. Comparisons between our proposed method and several alternative methods mentioned in Section 3.2. As highlighted in block font, our method successfully tackles the challenge of "task heterogeneity" and benefits performance by unifying more data.

Dataset	Method	Performance	
		Seg. (mIoU)	**Sal.** (mIoU)
Pascal-Context (5)	Single-Task FL	0.2449	0.5619
	Plain Many-Task FL	0.2731	0.5638
	FedProx	0.2745	0.5661
	Ours (DG w/ sc agg.)	**0.2762**	**0.5774**
Pascal-Context & NYUD (9)	Single-Task FL	0.2449	0.5619
	Plain Many-Task FL	0.2707	0.5714
	FedProx	0.2611	0.5745
	Ours (DG w/ sc agg.)	**0.2938**	**0.5814**

of our proposed method in terms of coping with data shortage. Note that the performance is supposed to drop when increasing the FL scale since the FL is more "decentralized" when less data is hosted on each client.

4.4. Improvements under Different K

K is utilized in TopK of dynamic grouping, which serves as the vital step to balance between using more diverse clients with different tasks and excluding the highly dis-

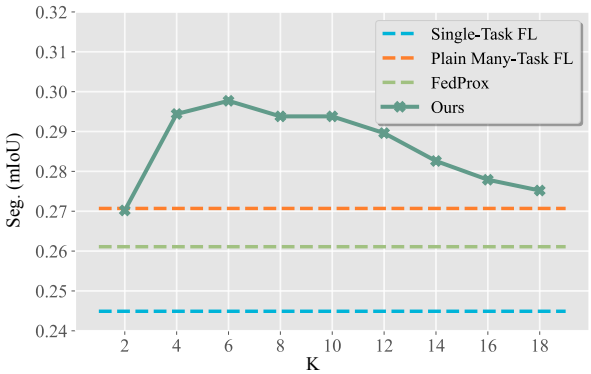

Figure 4. The performance measured by Seg. mIoU under different K, showing the effectiveness within the wide range of $4 \sim 18$. Note that K is utilized in the TopK step in the dynamic grouping. For every client's aggregation, a larger K means more other clients are "group" as its friends and be involved in aggregation.

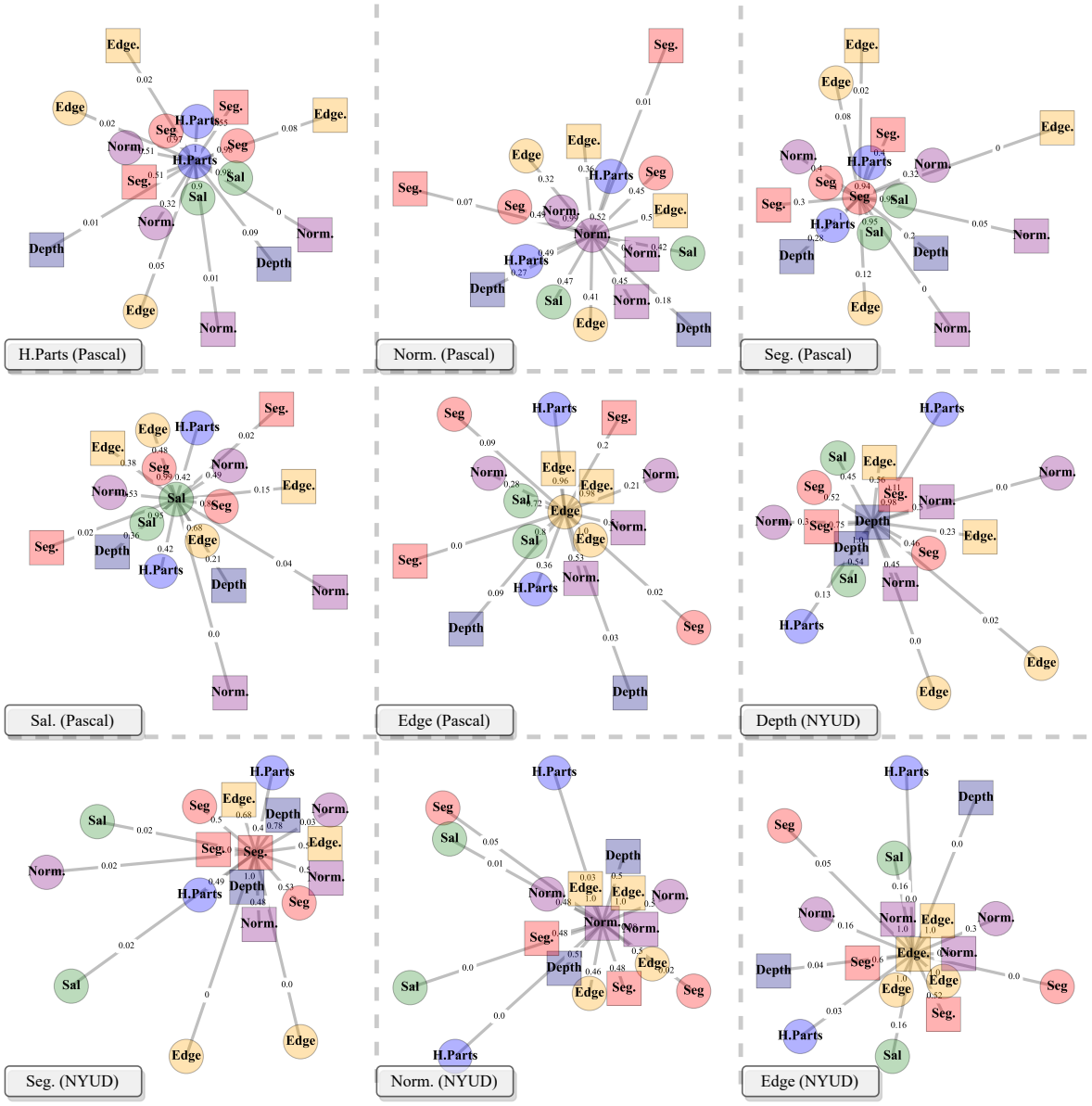

Figure 5. The visualization of average grouping results across the whole training process. For every client, we visualize the frequency of every other client being selected and included in its aggregation. For instance, in the first sub-figure which depicts the grouping result of clients specializing H.Parts, the edge between two clients performing H.Parts (represented by two blue circles) equals 1, which means 100 out of 100 times, the H.Parts client successfully identifies another H.Parts client and includes it in its aggregation. We use squares to represent NYUD clients while using circles to represent Pascal-Context Clients, and different colors represent different types of tasks.

similar ones. Therefore, it is important to analyze how different K affects performance. Here, we use the Pascal-Context(9) setting with the number of clients per task as 2, which means there are 18 clients in total. We compare our results with several alternative methods explored in Section 3.2 and provide results in Figure 4. Note that if we set $K = 2$ and every client successfully chooses the other client of the same task in each grouping step, the method will degenerate to "Single-Task FL". And "Plain Many-Task FL" is also a special case when we set $K = 18$ and every client is assigned with identical weights in the aggregation step. By changing K within the range of $2 \sim 18$, we plot the performance for each K using the Seg. mIoU as the metric. ❶ Our method shows superior performance than all of the baseline methods when K is within the large range of $4 \sim 18$, which shows our effectiveness is not sensitive to the choice of K. ❷ The performance drops if we use an extremely small K. That is because a small K (as small as 2)

Table 2. The effectiveness of our method compared to methods explored in Section 3.2 under different FL scales using Pascal-Context (5). Note that by default we keep the number of participating clients of every task to the same, which means (# Clients per task) $\times$ (# Tasks). equals the total number of clients. Therefore, the larger (# Clients per task) means a larger FL scale.

# Clients (per Task)	Method	Performance	
		Seg. (mIoU)	**Sal** (mIoU)
1	Single-Task FL	0.2896	0.5730
	Plain Many-Task FL	0.2738	0.5712
	FedProx	0.3032	0.5604
	Ours (DG w/ sc agg.)	**0.3045**	**0.5803**
2	Single-Task FL	0.2449	0.5619
	Plain Many-Task FL	0.2731	0.5638
	FedProx	0.2745	0.5661
	Ours (DG w/ sc agg.)	**0.2762**	**0.5774**

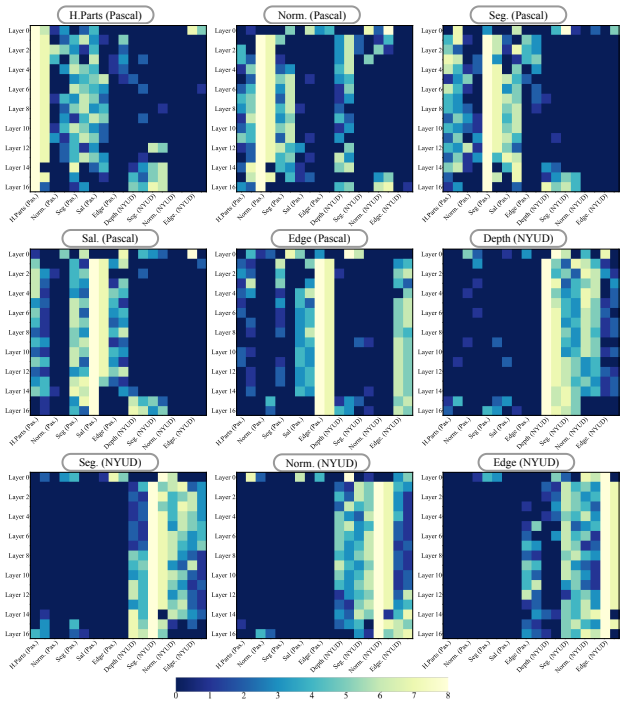

Figure 6. Illustration of every layer's voting. The brighter color means the related client get a higher vote, indicating the layer is more willing to aggregate the related client. Note that if perform TopK, each layer will assign all clients with voting values in descending order (the most similar one assigned with "K", the second most similar "K-1", ...).

will invalidate our method's capability to utilize more data, only weakening the method by imperfect grouping results.

5. Ablation Analysis

5.1. Grouping "Correctness"

The clients performing the same type of task do not have task heterogeneity, which means they are supposed to behave similarly and help each other thus to be chosen in the aggregation. Based on such prior knowledge, we define the grouping to be "correct" if the clients performing the same task are identified as "friends" and are included in the aggregation process. Therefore, it is important to check whether the dynamic grouping method successfully includes those clients who are known helpful in the training.

Based on the default Pascal-Context & NYUD(9) dataset where $N/C = 2, C = 9$ and $K = 8$, we analyze the grouping results in the whole training process in Figure 5. For every client, we calculate the frequency of every other client being included in its aggregation step, and we visualize it in the related sub-figure. Note that for each task, we randomly select one client and show its "correctness".

The results show that in the vast majority of cases, all clients almost perfectly choose and group other clients that share the same task. Besides, it validates our method would benefit training by enlarging the scope of data to be used for aggregation. For instance, as shown in the third sub-figure, in nearly half of the communication rounds, Seg.(Pascal-Context) clients would pick Seg.(NYUD) clients.

5.2. Voting Consistency

As is illustrated in Section 3.4, layers will first make their own choices on which client to aggregate with. Then the model trainer will consider opinions from all layers and make the final decision. We further analyze whether those layers agree with each other. Using the Pascal-Context & NYUD(9) setting, we present results of one client per task, and each sub-figure in Figure 6 shows the voting of a specific client. It is clear that a majority of layers would make similar choices, which means our method is not sensitive to the choice of layers for dynamic grouping. In all experiment, by default, we use the last 8 convolutional layers.

6. Conclusion and Discussions

In this work, we introduce the novel "many-task federated learning" (MaT-FL) setting to allow local clients performing different tasks to collaboratively train a model. This new setting brings potential benefits by unifying more data while raising the challenge of "task heterogeneity". To tackle the challenge, we design a dynamic grouping mechanism, enabling every client to dynamically choose and aggregate weights from its "friends". Extensive experiments also show its potential to scale FL up to **9 tasks, 2 data domains**). While this work is a pilot study, we hope for more attention from the community in this new setting.

References

[1] 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. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 5

[2] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 2, 3, 4

[3] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021. 3

[4] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020. 3

[5] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016. 2, 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[7] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7865–7873, 2021. 2, 3

[8] Ahmed Imteaj, Khandaker Mamun Ahmed, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. Federated learning for resource-constrained iot devices: Panoramas and state of the art. *Federated and Transfer Learning*, pages 7–27, 2023. 1, 2

[9] 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. 3

[10] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022. 1, 2

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020. 1, 2, 3

[12] Hanxue Liang, Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, and Zhangyang Wang. M^{3}vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. In *Advances in Neural Information Processing Systems*, 2022. 3

[13] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34:15434–15447, 2021. 3

[14] 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*, 2017. 1, 2, 3

[15] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016. 2, 3

[16] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014. 2, 4, 5

[17] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 4, 5

[18] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*, 2017. 2, 3

[19] Yasmin SarcheshmehPour, Yu Tian, Linli Zhang, and Alexander Jung. Networked federated multi-task learning. *arXiv preprint arXiv:2105.12769*, 2021. 3

[20] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017. 3

[21] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 4

[22] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021. 2, 3

[23] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33, 2020. 3

[24] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020. 2, 3, 4

[25] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*, 2020. 2, 3

[26] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 2

[27] 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, 2

[28] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Smart multi-tenant federated learning. *arXiv preprint arXiv:2207.04202*, 2022. 3