Learning to Collaborate in Decentralized Learning of Personalized Models

Shuangtong Li[1] Tianyi Zhou[2,3] Xinmei Tian[1] Dacheng Tao[4]
[1]University of Science and Technology of China; [2]University of Washington, Seattle
[3]University of Maryland, College Park; [4]JD Explore Academy
lst2015@mail.ustc.edu.cn, tianyizh@uw.edu, xinmei@ustc.edu.cn, dacheng.tao@gmail.com

Abstract

Learning personalized models for user-customized computer-vision tasks is challenging due to the limited private-data and computation available on each edge device. Decentralized learning (DL) can exploit the images distributed over devices on a network topology to train a global model but is not designed to train personalized models for different tasks or optimize the topology. Moreover, the mixing weights used to aggregate neighbors' gradient messages in DL can be sub-optimal for personalization since they are not adaptive to different nodes/tasks and learning stages. In this paper, we dynamically update the mixing-weights to improve the personalized model for each node's task and meanwhile learn a sparse topology to reduce communication costs. Our first approach, "learning to collaborate (L2C)", directly optimizes the mixing weights to minimize the local validation loss per node for a pre-defined set of nodes/tasks. In order to produce mixing weights for new nodes or tasks, we further develop "meta-L2C", which learns an attention mechanism to automatically assign mixing weights by comparing two nodes' model updates. We evaluate both methods on diverse benchmarks and experimental settings for image classification. Thorough comparisons to both classical and recent methods for IID/non-IID decentralized and federated learning demonstrate our method's advantages in identifying collaborators among nodes, learning sparse topology, and producing better personalized models with low communication and computational cost.

1. Introduction

Training large models for computer-vision tasks, like deep neural networks (DNNs) and vision transformers [6, 38, 40], is known to be data hungry but data in many applications are distributed over millions or even billions of personal/IoT devices. Transmitting their local data is usually forbidden due to privacy protection and limited communication bandwidth, so centralized learning on all data is infeasible. Instead, federated learning (FL) [29] and decentralized learning (DL) [24], as two widely studied distributed learning schemes, aim at training a global model by only sharing local models/gradients across multiple devices without leaking their private data. Both FL and DL iterates between on-device local model updates and cross-device model aggregation, i.e., each node uses its local data to update its model for iterations before updated by aggregating the models of other nodes. Their main difference is that FL periodically updates a global model on a central server by aggregating local models and synchronizes all nodes with the updated global model. In contrast, DL neither presumes a central server nor an explicit global model: each device can only communicate with its neighbor nodes on a network topology and aggregate their messages to update its own local model without global synchronization. Under certain assumptions to the topology and mixing weights for the aggregation [24], the local models in DL provably converge to a "consensus" global model. FL and DL target different

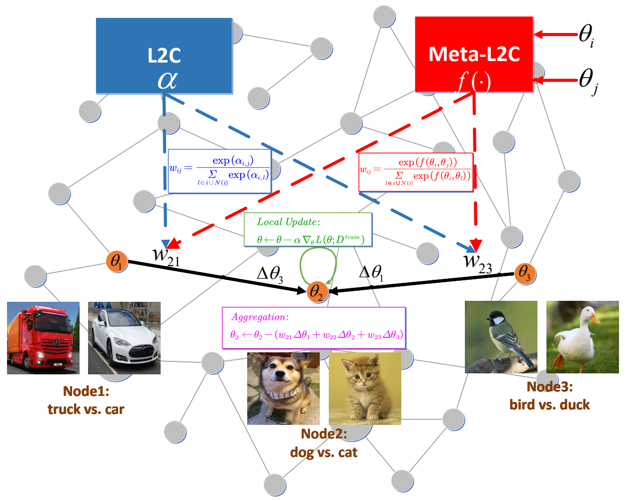

L2C
α
Meta-L2C
$f(\cdot)$
θ_i
θ_j
$w_{ij} = \frac{\exp(\alpha_{i,j})}{\sum_{l \in i \cup N(i)} \exp(\alpha_{i,l})}$
$w_{ij} = \frac{\exp(f(\theta_i,\theta_j))}{\sum_{l \in i \cup N(i)} \exp(f(\theta_i,\theta_l))}$
Local Update:
$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta; D^{train})$
w_{21}
w_{23}
θ_1
$\Delta\theta_3$
θ_2
$\Delta\theta_1$
θ_3
Aggregation:
$\theta_2 \leftarrow \theta_2 - (w_{21}\Delta\theta_1 + w_{22}\Delta\theta_2 + w_{23}\Delta\theta_3)$
Node1:
truck vs. car
Node2:
dog vs. cat
Node3:
bird vs. duck

Figure 1. Decentralized learning alternates between local model update and aggregation of neighbors' model updates. We propose "learning to collaborate (L2C)" and meta-L2C to optimize and produce the mixing weights $w_{i,j}$ for model aggregation in each round to achieve better personalization of local models on their own tasks/data. While L2C directly optimizes the mixing weights, meta-L2C learns an attention mechanism to automatically produce mixing weights between two models θ_i and θ_j.

distributed learning settings in practice.

However, in practice, devices and their users may target different tasks and their local data distributions are not identical, e.g., when they belong to separate geographic groups, so one global model may not perform well on all devices. How to personalize each local model for its own task/distribution and meanwhile exploit the knowledge shared across devices is a key challenge emerging in both FL and DL. Recently, there has been a growing interest to address the data/task heterogeneity across devices by personalization [5, 8, 20, 32, 33]. In FL, since the global model is explicitly optimized and broadcasted, the trade-off between global consensus and local model personalization is inevitable and may rely on careful hyperparameter tuning [20]. It becomes even more challenging when taking other criteria, e.g., fairness and robustness of the global model, into account. On the contrary, it is more natural to formulate local personalization in DL: the main challenge is to optimize the aggregation of neighbors' messages for each device's own task instead of the global consensus. Recently, CGA [7] considers learnable mixing weights for each node in non-IID DL and finds a descent direction positively correlated with its neighbors' gradients. However, this does not directly optimize the personalization performance and introduces an extra constraint. Moreover, the aggregation is only learned for a fixed set of nodes/tasks on a predefined topology and thus cannot generalize to newly added devices or unseen tasks.

In this paper, we take a further step towards more automated and adaptive decentralized learning of personalized local models. To this end, we learn an aggregation scheme, i.e., "learn to collaborate (L2C)", which automatically weighs and mixes neighbors' messages to update each local model for better performance on its local task/data. Similar to FL/DL, we alternate between local model training and the aggregation of neighbors' model updates for each node. Inspired by meta-learning, we formulate this problem as minimizing the validation loss of each local model after being updated by the weighted aggregation per round. L2C optimizes the mixing weights solely based on the validation loss so every node is agnostic to other nodes' tasks or data distributions for better protection of their privacy. Empirically, the mixing weights learned by L2C can faithfully reflect whether two nodes have similar tasks and/or data distributions and thus are usually sparse in practice. Moreover, they quickly converge in earlier stages so we can leverage their sparsity to create a sparse network topology, which can substantially save the communication cost for the rest training.

However, the mixing weights learned by L2C are associated with a fixed set of pre-defined nodes/tasks and thus cannot be adapted to new nodes/tasks or data distributions. To avoid optimizing the mixing weights from scratch for new nodes, we propose meta-L2C that learns an attention mechanism to automatically assign the mixing weights given the models of a node and its neighbors. Similar to meta-learning, we can train meta-L2C on a training set of nodes with different tasks and then apply it to a new set of nodes/tasks without re-training or fine-tuning. Hence, a pre-trained meta-L2C can produce mixing weight with sparse topology that accurately capture the task similarity at the very beginning of training, which further improves the training and communication efficiency of L2C.

We summarize the main framework of L2C and meta-L2C in Figure. 1, both only requiring light-weight models. In experiments on three benchmark datasets, L2C and meta-L2C consistently outperform FL/DL methods with and without specific designs for personalization or data heterogeneity. We further demonstrate the promising generalization performance of meta-L2C when transferred to unseen tasks or data. Moreover, we present an empirical analysis with case studies to show that L2C/meta-L2C can produce mixing weights precisely capturing the task correlation among nodes and a sparse topology for efficient communication. In addition, we empirically analyze how sensitive L2C/meta-L2C is to training/validation splitting and communication rounds/budget.

2. Related Work

Federated learning (FL) [29] trains a global model over various nodes/clients by alternating between local (stochastic) gradient descent on each node and aggregation of local models on the server side as an update to the global model. Previous studies find that the performance of FL degrades in the non-IID setting when data distributions on devices are not identical, both empirically [13] and theoretically [16]. Moreover, the statistical heterogeneity weakens the global model's performance on some nodes' local tasks and results in poor fairness across nodes [22]. Several strategies have been studied to address this non-IID challenge: (1) modifying the model aggregation, e.g., by knowledge distillation [27], clustering and sampling low-variance updates for aggregation [10], Bayesian reformulation [4], matching the neurons/channels of different models before aggregation [35], selecting a diverse subset of clients with representative gradient information [2], etc. (2) Regularizing the local objectives with proximal terms [1, 21]. These methods focus on improving FL of a global model to be more robust to non-IID distributions but they are not designed to produce a personalized model for each node, which is a open challenge attracting increasing interests in recent literature. When each node does not have sufficient data for training a personalized model alone, it can still leverage the knowledge of other nodes via the global model in FL. A variety of methods explicitly optimize a personalized model per node in FL: (1) optimization formulation that performs a trade-off between the global model and local personalization [20, 33]; (2) clustering of local nodes [11, 31, 37] and applying aggregation within each cluster; (3) personalizing some layers of local models, i.e., batch normalization layers [23], shallow layers [25], local heads [5]; (4) distillation of global-view knowledge of data to the local model training [42]; (5) training the global model as an initialization [8] or an generator [32] of local models, or (6) sending prototypes [34] to regularize the training of local models. Most of methods train a global model shared by all nodes and meanwhile optimizes a local model per node, so it is almost inevitable to perform a trade-off between the global and local objectives, leading to sub-optimal personalized models for local clients' data distributions.

Decentralized learning does not presume a central server or an explicit global model aggregation, though its goal is the consensus of all local models towards the same model. Earlier works combine the gossip-averaging [3] with SGD. Under assumptions of the topology like doubly stochastic mixing-weights [15], all local models can be proved to converge to a "consensus model" [24] after iterating peer-to-peer communication. Although they show promising performance in the IID setting, [13] points out that they suffer from severe performance degeneration in non-IID scenarios. To tackle this problem, [26] modifies the momentum term of decentralized SGD to be adaptive to heterogeneous data; [13] replaces batch normalization with layer normalization to avoid the divergence of models. [17] assign nodes to clusters but still need to assume those within each cluster share the same task and distribution, which may not hold in general heterogeneous settings. [7] finds a descent direction per node close to its local gradient and meanwhile positively correlated with its neighbors' gradients, which removes the conflicts caused by heterogeneity. These methods still focus on achieving a global consensus model, which can be sub-optimal to each local task. Moreover, various methods require to communicate and aggregate the local gradients frequently (e.g., every step), while FL allows multiple local epochs between aggregations.

Since many works in FL/DL above aim at training a global model or reaching a consensus over all nodes, local personalization in them has to perform trade-off between the global objective and local tasks. In contrast, we solely focus on personalizing local models in the DL setting and study how to aggregate neighbors' gradients to improve each node's model for its own task and data distribution. Inspired by meta-learning such as MAML [9], we explore the idea of "learning to collaborate" by finding the mixing weights of neighbors minimizing the validation loss on each node, which shares similar intuition as relating different tasks on graph in meta-learning [28] and multi-task learning [41]. Comparing to DL methods adopting MAML idea, e.g., which trains a meta model as an initialization [8] or an generator [32] of local models, our meta model is the mixing weight matrix or its generator, which is much more light-weight, easier to train, and generalizes better.

3. Leaning to Collaborate

In this section, we develop a decentralized learning approach "learning to collaborate (L2C)" that updates each node's personalized model by weighted aggregation of its neighbors' model updates. Similar to FL/DL, L2C alternates between local updates and model aggregation. Unlike FL/DL, the mixing weights of aggregation in L2C are learnable and dynamic rather than pre-defined and fixed. Moreover, each node collaborates with its neighbors solely for improving its local model so it does not need to perform trade-off between global consensus and personalization. We formulate the problem as minimizing the validation loss of the personalized model on each node. While L2C directly optimizes the mixing weights for a fixed set of nodes/tasks, we further propose meta-L2C that can automatically assign mixing weights given models of nodes/tasks by an attention mechanism. We elaborate on algorithms of L2C and meta-L2C, followed by an analysis of their communication and computational complexity.

3.1. Decentralized Learning of Personalized Models

We study the non-IID setting for K local nodes/devices and each node $i \in [K]$ has its own task defined on a data distribution $\mathcal{D}_i$, which differs from those for other nodes. A private dataset $\boldsymbol{D}_i$ is available at each node i, which can be split into a training set $\boldsymbol{D}_i^{train}$ and a validation set $\boldsymbol{D}_i^{val}$. In various practical scenarios, each node does not have sufficient data to train a reliable local model solely by itself. It cannot leverage data of other nodes due to privacy concerns but communicating the models and gradients are usually allowed. By following a similar protocol in distributed learning, we alternate between local (stochastic) gradient steps on each node and an aggregation step of model updates from neighbor nodes. Specifically, at round-t, the local learning at each device i starts from θ_i^t and runs s steps of gradient descent to minimize its training loss $\mathcal{L}(\theta_i; \boldsymbol{D}_i^{train})$, i.e.,

$$\begin{aligned} &\text{Initialize} \quad \theta_i^{t+\frac{1}{2}} \leftarrow \theta_i^t, \\ &s \text{ steps of} \quad \theta_i^{t+\frac{1}{2}} \leftarrow \theta_i^{t+\frac{1}{2}} - \alpha \nabla_\theta \mathcal{L}(\theta_i^{t+\frac{1}{2}}; \boldsymbol{D}_i^{train}), \end{aligned} \tag{1}$$

followed by an aggregation step that updates $\theta_i^{t+\frac{1}{2}}$ with a combination of model updates $\Delta\theta_j^t \triangleq \theta_j^t - \theta_j^{t+\frac{1}{2}}$ sent from its neighbor nodes $\mathcal{N}(i)$, i.e.,

$$\begin{aligned} \text{Aggregation} \quad \theta_i^{t+1} &= \theta_i^{t+\frac{1}{2}} - \sum_{j \in \mathcal{N}(i)} w_{i,j} \Delta\theta_j^t \\ &= \theta_i^t - \sum_{j \in i \cup \mathcal{N}(i)} w_{i,j} \Delta\theta_j^t, \end{aligned} \tag{2}$$

where the mixing weight $w_{i,j}$ can be explained as a "collaboration score" of node-j for node-i.

3.2. "Learning to Collaborate" with Learnable Mixing-Weights

Unlike the mixing weights $w_{i,j}$ pre-defined in FL or DL, whose goal is the convergence of θ_i^t for all client $i \in [K]$ towards a consensus global model under an IID distribution, we train the mixing weights for better personalization performance of local models on their own data distributions. In particular, our goal is to find mixing weights resulting in an aggregated model θ_i^{t+1} minimizing the validation loss on $\boldsymbol{D}_i^{val}$ for every client $i \in [K]$, i.e.,

$$\min_{\alpha_i} \mathcal{L}(\theta_i^{t+1}; \boldsymbol{D}_i^{val}), \;\; w_{i,j} = \frac{\exp(\alpha_{i,j})}{\sum_{\ell \in i \cup \mathcal{N}(i)} \exp(\alpha_{i,\ell})}, \quad (3)$$

where the mixing weights in w_i are computed from learnable parameters α_i by the softmax function. Intuitively, $w_{i,j}$ should be large if the tasks and data distributions of client-i and client-j are similar. However, in various practical settings, this similarity is usually unknown a priori and is hard to estimate without data sharing or an accurate similarity metric. Hence, we propose a "learning to collaborate (L2C)" framework that learns the mixing weights in an end-to-end manner. In particular, L2C alternates between updating/aggregating the local models (i.e., Eq. (1)-(2)) and optimizing the mixing weights (i.e., Eq. (3)). Thereby, the mixing weights can be trained to adapt to model aggregation in multiple rounds and converge to values precisely reflecting the "collaboration score" between clients. Comparing to the training cost of FL/DL, we only have K^2 additional parameters to optimize, so L2C does not introduce any significant extra computation per round.

3.3. "Learning to Collaborate" as a Meta-Learner

Despite L2C's simplicity in parameterizing the mixing weights, it lacks the flexibility and capability of producing adaptive mixing weights for different training stages or new clients with unseen tasks. For example, the most similar neighbors of a client might be assigned with the largest mixing weights during earlier stages but their resulted improvements might diminish later due to over-exploitation. On the other hand, it is common in practice to add new nodes with unseen tasks on the fly to the network, so the mixing weights optimized for a fixed set of clients cannot be generalized to those new clients. Inspired by the idea of meta-learning, we further study a meta-L2C model that learns to produce mixing weights adaptive to training stages, new client models, or unseen tasks. In particular, meta-L2C is composed of an encoder of model weights and an attention module, where the former produces compact representations of local models and the latter computes the mixing weights from pairwise similarity between the representations.

In particular, we compute the mixing weight $w_{i,j}$ from a learnable similarity $f(\theta_i, \theta_j)$ between θ_i and θ_j, i.e.,

$$w_{i,j} = \frac{\exp(f(\theta_i, \theta_j))}{\sum_{\ell \in i \cup \mathcal{N}(i)} \exp(f(\theta_i, \theta_\ell))}, \quad (4)$$

Following the definition of dot-product attention, $f(\theta_i, \theta_j)$ can be computed as the inner product between the representations of θ_i and θ_j produced by the encoder $E(\cdot; \alpha)$. Specifically, at round-t, after the local updates in Eq. (1) and before the aggregation in Eq. (2), we compute the representation of θ_i as $E(\theta_i^{t+\frac{1}{2}} - \theta_i^0; \alpha)$, which depends on the change of θ_i since its initialization θ_i^0. We remove θ_i^0 so the encoder's input is mainly determined by the (stochastic) gradients computed on the data over the past training rounds. Hence, $f(\theta_i, \theta_j)$ at round-t is defined as

$$f(\theta_i, \theta_j) = \langle E(\theta_i^{t+\frac{1}{2}} - \theta_i^0; \alpha), E(\theta_j^{t+\frac{1}{2}} - \theta_j^0; \alpha) \rangle. \quad (5)$$

We then update each local model by aggregating its neighbors' model updates (Eq. (2)) with mixing weights produced by the attention module in Eq. (4). Similar to L2C, we minimize the validation loss of the aggregated models $\{\theta_i^t\}_{i=1}^K$ to train the meta-L2C model, i.e.,

$$\min_{E(\cdot;\alpha)} \frac{1}{K} \sum_{i=1}^{K} \mathcal{L}(\theta_i^{t+1}; \boldsymbol{D}_i^{val}). \quad (6)$$

In practice, DNNs usually contain millions to billions of parameters, which can significantly increase the input dimension of $E(\cdot; \alpha)$ and its model size. For structured DNNs such as convolutional neural networks, we can adopt a channel-sharing strategy and apply the same encoder to each channel's parameters from the same layer. Specifically, we apply $E_l(\cdot)$, e.g., a two-layer fully-connected network, to each filter in layer-l and concatenate the output embedding for all the C filters in the same layer, i.e.,

$$\phi_l = [E_l(\theta_{l,1}), E_l(\theta_{l,2}), \cdots, E_l(\theta_{l,C})], \quad (7)$$

where $\theta_{l,j}$ is the parameters of filter/channel-j at layer-l. The representation of the whole model $E(\theta)$ concatenates the embedding for all the L layers, i.e.,

$$E(\theta) = [\phi_1, \phi_2, \cdots, \phi_L]. \quad (8)$$

3.4. Algorithms of L2C and meta-L2C

We present the complete algorithms for L2C and meta-L2C in Algorithm 1: they share most procedures except the highlighted steps. Each algorithm alternates between local model updates (line 6-11) and model aggregation (line 15-16), where the aggregation weights are produced by L2C or meta-L2C in line 13-14. We address the optimization problems in Eq. (3) and Eq. (6) by (stochastic) gradient descent in line 18. At the end of the first T_0 rounds, we try to remove the top K_0 neighbors with the smallest mixing weights for each node in order to create a sparse topology saving communication costs for future training rounds.

Algorithm 1: L2C and meta-L2C

```
1  Input $\alpha, \beta$, neighbour sets $\{\mathcal{N}(i)\}_{i=1}^K$,
     $\{D_i^{train}\}_{i=1}^K, \{D_i^{val}\}_{i=1}^K, S, T, T_0, K_0$
2  Output $\{\theta_i^T\}_{i=1}^K$
3  Initialize $\{\theta_i^0\}_{i=1}^K$ and $\{\alpha_i\}_{i=1}^K$
4  for $t = 0 : T$ do
5      for device $i = 1 : K$ in parallel do
6          // Local SGD updates
7          $\theta_i^{t+\frac{1}{2}} \leftarrow \theta_i^t$
8          for local SGD step $m = 0 : S$ do
9              $\theta_i^{t+\frac{1}{2}} \leftarrow \theta_i^{t+\frac{1}{2}} - \alpha \nabla_\theta \mathcal{L}(\theta_i^{t+\frac{1}{2}}; D_i^{train})$
10         end
11         $\Delta\theta_i^t \leftarrow \theta_i^t - \theta_i^{t+\frac{1}{2}}$, $\Delta\widehat{\theta}_i^t \leftarrow \theta_i^{t+\frac{1}{2}} - \theta_i^0$
12         // Mixing weights calculation
13         $w_{i,j} \leftarrow \frac{\exp(\alpha_{i,j})}{\sum_{\ell \in i \cup \mathcal{N}(i)} \exp(\alpha_{i,\ell})}$
14         $w_{i,j} \leftarrow \frac{\exp(f(\Delta\widehat{\theta}_i^t, \Delta\widehat{\theta}_j^t))}{\sum_{\ell \in i \cup \mathcal{N}(i)} \exp(f(\Delta\widehat{\theta}_i^t, \Delta\widehat{\theta}_\ell^t))}$
15         // Aggregation
16         $\theta_i^{t+1} \leftarrow \theta_i^t - \sum_{j \in \mathcal{N}(i)} w_{i,j} \Delta\theta_j^t$
17         // Update L2C/meta-L2C parameters
18         $\alpha_i \leftarrow \alpha_i - \beta \nabla_{\alpha_i} \mathcal{L}(\theta_i^{t+1}; D_i^{val})$
19         // Remove edges for sparse topology
20         if $t == T_0$ then
21             For each device $i$, remove $K_0$ neighbors
               $j \in N(i)$ with the smallest $w_{i,j}$
22         end
23     end
24     $\alpha_i \leftarrow AllReduce(\alpha_i)$
25 end
```

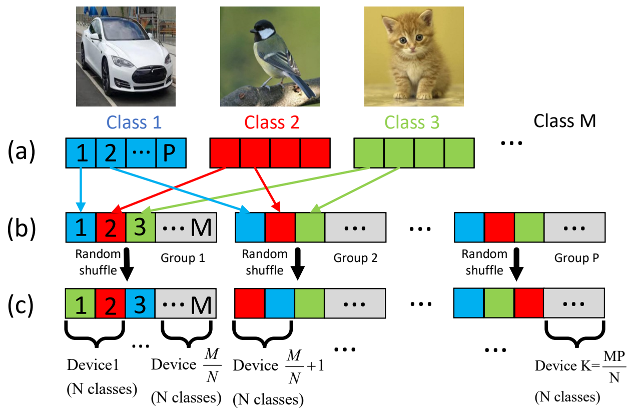


Figure 2. The generation of non-IID tasks in our experiments. Given a dataset of M classes each having P shards of data, our goal is to draw $K = MP/N$ tasks, each defined on $N < M$ random classes with one random shard of data per class. (a) The data of each class is partitioned into P shards. (b) We randomly draw one shard from every class to form a group of M shards. Repeating this process yields P groups. (c) We randomly shuffle the order of shards in each group and then assign every N shards from left to right to a device.

Communication cost The communication cost of L2C is $\mathcal{O}(m_s N_b)$, where m_s is the number of edges on the topology and N_b is the model size, so the sparse topology in line 21 reduces the cost. On the other hand, meta-L2C needs extra communication cost for sending/receiving $\Delta\widehat{\theta}_i$ in line 11 and for AllReduce in line 24 of Algorithm 1: the former's cost is $\mathcal{O}(m_s N_b)$, while the latter's cost can be much smaller if the average number of channels per layer $\bar{c}$ is large and the output dimension d of $E_l(\cdot)$ is small, e.g., for Ring-AllReduce [12] across K nodes, the cost is $\frac{2(K-1)dN_b}{\bar{c}}$. Hence, we can have similar communication cost $\mathcal{O}(m_s N_b)$ as existing DL methods, e.g., DPSGD [24] and CGA [7]. However, this might be an limitation if the above conditions are violated.

Computational cost Comparing to existing DL methods, the additional computation required by our method is line 18. However, its computation can be negligible compared to the cost of local updates when we apply a large S in line 8. In our experiments, we apply 5 epochs of local updates in each round and achieve promising personalization performance.

4. Experiments

Table 1. Non-IID tasks randomly generated by Fig. 2 for the three datasets.

Dataset	M	P	K	N	model
CIFAR-10	10	20	100	2	two-layer CNN [29]
CIFAR-100	100	10	100	10	two-layer CNN [29]
MiniImageNet	100	10	100	10	four-layer CNN [9]

Experimental Setup Inspired by the non-IID setting used in FedAvg [29], we randomly assign different classification tasks and the associated training data to each node. In most experiments, we generate these non-IID tasks by the procedure detailed in Fig. 2 unless specified otherwise. We evaluate L2C and meta-L2C and compare them with several FL/DL baselines on three datasets: **CIFAR-10** [19], **CIFAR-100**, and **MiniImageNet** [30], each having 50,000 training images and 10,000 test images. In Table 1, we list the parameters in Fig. 2 for each dataset. For each node, we randomly select 80% of its local data as the training set and leave the rest as the validation set. Following the evaluation setting for personalized models in previous non-IID DL works [25, 39], we evaluate each local model on all the available test data belonging to the classes in its local task. In order to report the mean and standard deviation of the evaluated accuracy, we run each experiment on five random non-IID partitions produced by Fig. 2. For local models, we choose the two-layer CNN adopted in FedAvg [29] for CIFAR-10/100 and the four-layer CNN adopted in MAML [9]. Since batch-norm may have a detrimental effect on DL [13], we replace all the batch-norm layers [14] with group-norm layers [36].

Implementation of L2C and meta-L2C models We apply a lightweight fully connected network of two layers with

Method	CIFAR-10	CIFAR-100	MiniImageNet
Local SGD only	87.50±1.37	55.47±2.08	41.59±2.56
DPSGD(s=1 step)	83.01±1.31	40.56±1.24	30.26±2.39
DPSGD(s=5 epochs)	75.89±1.44	35.03±1.39	28.41±2.23
CGA(s=1 step)	65.65±2.37	30.81±3.82	27.65±3.10
CGA(s=5 epochs)	diverge	diverge	diverge
FedAvg	70.65±3.88	40.15±4.01	34.26±4.44
FOMO	88.72±0.29	52.44±0.70	44.56±0.77
Ditto	87.32±0.69	54.28±0.57	42.73±1.21
L2C(ours)	90.14±0.34	**59.00±0.42**	**50.03±0.75**
meta-L2C(ours)	**92.10±0.72**	58.28±1.26	48.80±1.42

Table 2. Test accuracy (mean±std) of 100 local models on non-IID tasks produced by Fig. 2. L2C and meta-L2C outperform all FL/DL baselines.

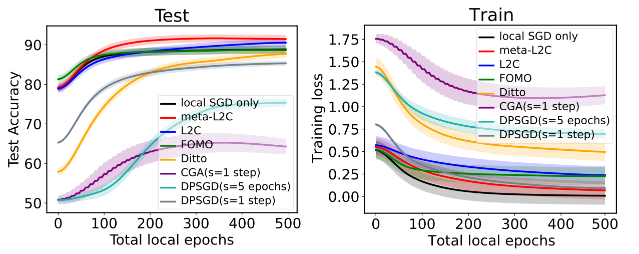


Figure 3. Test accuracy and training loss (mean±std) vs. total local epochs on CIFAR-10. L2C and meta-L2C converge faster to better test/training performance than FL/DL baselines. All methods run for 100 communication rounds except CGA(s=1 step) and DPSGD(s=1 step) which communicate per local SGD step. FedAvg requires $>$ 1000 epochs to converge and is not included.

output dimensions $(10, 5)$ as the encoder $E_l(\cdot)$ in meta-L2C. We use Adam [18] with learning rate of 0.1 and weight decay of 0.01 to train both L2C and meta-L2C. We apply Algorithm 1 and train all the local models for T=100 rounds with 5 epochs of local SGD per round. To achieve a sparse topology, we remove 90% (i.e., $K_0 = 90$) of neighbors for each node after $T_0 = 10$ rounds.

Baselines We compare our methods with a diverse set of baselines from federated learning (FL) and decentralized learning (DL) literature, as well as a **local SGD only baseline** without any model aggregation across nodes. FL baselines include **FedAvg** [29] (the most widely studied FL method), **Ditto** [20] achieving fairness and robustness via trade-off between the global model and local objectives, and **FOMO** [39] training personalized models only with adaptive mixing weights. DL baselines include the commonly used **DPSGD** [24] with fixed mixing weights and topology, and **CGA** [7] with a fixed topology but adaptive mixing weights for removing the conflict of cross-device gradients.

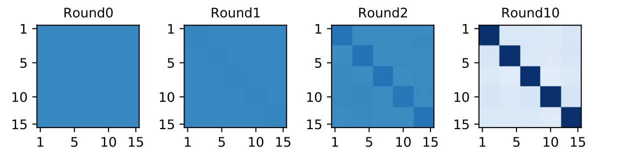
We run each baseline for 100 (communication) rounds or equally 500 local epochs (if needing $>$ 100 rounds), the same as our methods, except FedAvg which needs more (i.e., $>$ 1000) epochs to converge. For fair comparisons, we keep their communication cost per node and local epochs in each round to be no smaller than that of our methods. For FL baselines, the communication happens between the global server and clients, so we randomly select 10% clients for aggregation and apply 5 local epochs per client in each round. For DL baselines, i.e., DPSGD and CGA, we let every device communicate with about 10% nodes in every round. Since they originally propose to only run one local SGD step per round on a single mini-batch, we evaluate them with two settings, i.e., one local step per round and 5 local epochs per round, and we apply more rounds for the former to match the total local epochs (i.e., 500 epochs) of other methods. To match the communication cost of our methods, we extend the ring and bipartite topology used in previous DL works [7] to increase the number of neighbors for each node. Specifically, we study (1) a "group-ring" topology that connects two nodes i and j if $|i - j| \leq \frac{(K-K_0)}{2}$ or $K - |i - j| \leq \frac{(K-K_0)}{2}$; and (2) a generalized bipartite topology that randomly partition all nodes into two groups and then connect each node in a group to $K - K_0 = 10$ nodes randomly drawn from the other group. In our experiments, they both outperform their original versions with fewer neighbors and communications. Hence, in the following, we always report the best result among all the four types of topology for each DL baseline.

Training Hyperparameters In all methods, for local model training, we use SGD with learning rate of 0.01, weight decay of 5×10^{-4}, and batch size of 10. For other hyperparameters of baselines, we use the values proposed in their papers except the learning rate, which is kept constant and is tuned/selected as the best validation accuracy among $[0.01, 0.05, 0.1]$.

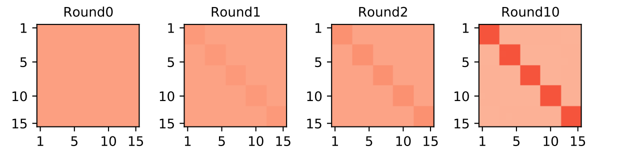
4.1. Main Results

Local test accuracy and convergence In Table 2, we report the test accuracy of all the 100 nodes' models on their assigned non-IID tasks. L2C and meta-L2C outperform all the FL/DL baselines by a large margin over all the three datasets and different random non-IID partitions. For the baselines, we notice that those solely targeting a global consensus model without encouraging personalization of local models, e.g., FedAvg, DPSGD, and CGA, can perform even worse than the simple "local SGD only" method that solely focuses on personalization. Ditto, which performs a trade-off between global consensus and local objectives, and FOMO, which shares a similar personalization objective (local validation loss) as ours, perform better than the other baselines but still worse than our methods. In Fig. 3, we show the convergence of test accuracy and training loss for all methods: although FOMO and local SGD only also converge fast, meta-L2C achieves the fastest convergence on the test accuracy.

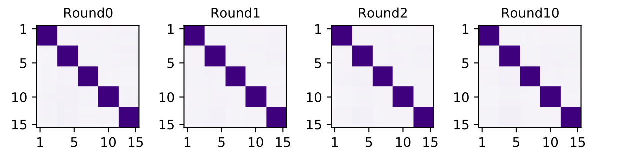
Learned mixing weights and topology In Fig. 4a-4b, we report how the mixing weights produced by L2C and meta-L2C over communication rounds for a simpler non-IID setting that there exist groups of nodes assigned with identical tasks. The results show that our methods can quickly identify nodes of the same task and assign larger mixing weights to them in aggregation, which produces better personalized



(a) L2C

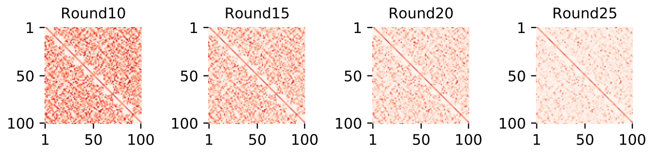


(b) meta-L2C

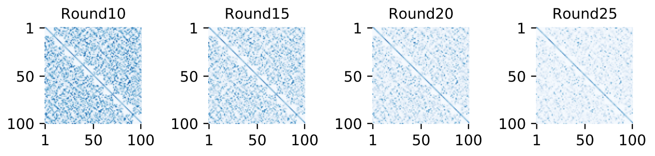


(c) Transfer meta-L2C: CIFAR-100 → CIFAR-10

Figure 4. Mixing weights of 15 nodes produced by (a) L2C, (b) meta-L2C, and (c) a meta-L2C pretrained on CIFAR-100 tasks, at different rounds on CIFAR-10. For simplicity, instead of Fig. 2, we randomly partition the 10 classes into five two-class tasks and assign every task to three nodes, each getting $1/3$ of the task's data. So node 1-3 shares task-1, node 4-6 shares task-2, etc. Both L2C and meta-L2C can produce mixing weights identifying the nodes with the same task after 10 rounds, while meta-L2C pretrained on a dataset's tasks can generalize to unseen tasks and produce accurate mixing weights after the first round.



(a) L2C

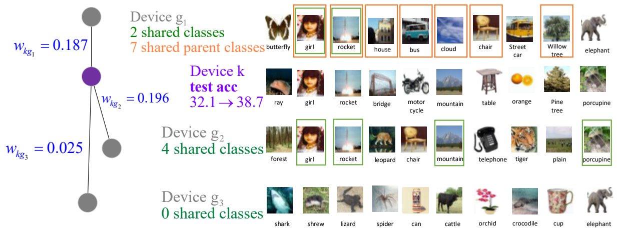


(b) meta-L2C

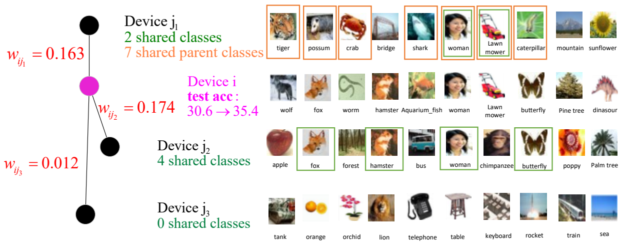
Figure 5. Mixing weights of 100 nodes produced by (a) L2C and (b) meta-L2C on CIFAR-100. Their non-IID tasks are produced by Fig. 2 so nodes sharing exactly the same task are rare. However, nodes with shared classes can still collaborate to improve their own local models, and our methods can automatically discover a sparse topology of them.

models for the task. In Fig.5, we show a general case when there are few nodes with identical tasks. However, they may still share different number of classes and our methods are capable to learn a sparse topology to reflect such complicated task correlations after a few rounds. As mentioned before, a sparse topology can significantly reduce the communication cost of the later-stage training.

Fairness across nodes Fairness is a critical metric to evaluate whether all the local models performs equally good or drastically different on their own tasks. In Table 3, we report the test accuracy averaged over the worst 10% nodes



(a) L2C



(b) meta-L2C

Figure 6. A node (k or i) and its three neighbors with mixing weights produced by (a) L2C and (b) meta-L2C on CIFAR-100. In both cases, the neighbor nodes with many shared classes (g_2 or j_2) or with semantically similar but non-identical classes (e.g., g_1 or j_1 sharing many parent classes) are assigned with larger mixing weights, compared to neighbors without any shared or parent classes (g_3 or j_3). This explains the improvement of local test accuracy over the local SGD only baseline without any collaboration with neighbors.

and worst 20% nodes, as well as its mean and standard deviation over all nodes. Our methods achieve higher test accuracy on the worst performed nodes and keep a relative small variance among all nodes, indicating their advantage on encouraging fairness of the personalization performance across different nodes/tasks.

Table 3. Statistics of the test accuracy for non-IID tasks over 100 nodes on CIFAR-10. For each metric, we report its mean±std over five non-IID partitions randomly generated by Fig. 2. The "worst 10%", "worst 20%", and "std (standard deviation)" metrics reflect the advantage of L2C/meta-L2C on the fairness of persoanlization performance across nodes.

Method	**Mean**	**Worst 10%**	**Worst 20%**	**std**
Local SGD only	87.50±1.37	77.35±2.59	81.59±10.03	8.14±5.61
FOMO	88.72±0.29	78.80±10.00	85.20±4.96	9.02±5.72
L2C(ours)	90.14±0.34	**83.02±4.99**	85.10±0.36	**5.85±3.31**
meta-L2C(ours)	**92.10±0.72**	82.90±2.49	**88.84±0.21**	7.34±4.15

Generalization of pretrained meta-L2C to new tasks Compared to L2C, a key advantage of meta-L2C is that it can be adapted to new nodes and new tasks without training the mixing weights from scratch. We evaluate its generalization via transferring a meta-L2C model pretrained on a dataset's non-IID tasks (randomly generated by Fig 2) to unseen tasks from another dataset. We fix the meta-L2C model without fine-tuning when deployed to the unseen tasks. We summarize the transfer learning results between CIFAR-10 and CIFAR-100 in Table 4, which are comparable with the model trained on the same dataset and thus still outperform all the baselines. Moreover, we show the mixing weights of CIFAR-10 tasks in Fig. 4 produced by a meta-L2C pretrained on CIFAR-100 tasks in different rounds: in Fig. 4c, it produces the accurate mixing weights since the first round and thus can effectively improve the convergence and communication/computational efficiency.

Table 4. Generalization of a meta-L2C pre-trained on a dataset's tasks to unseen tasks of another dataset. The second row reports the test accuracy (mean±std) achieved by a meta-L2C pretrained on CIFAR-100(CIFAR-10) tasks and then applied to CIFAR-10(CIFAR-100) tasks. The meta-L2C transferred from another dataset can achieve comparable performance as the one trained on the same dataset's tasks.

Method	CIFAR-10	CIFAR-100
meta-L2C(pretrain)	92.10±0.72	58.28±1.26
meta-L2C(transfer)	91.33±0.89	56.59±1.40

4.2. Empirical Analysis

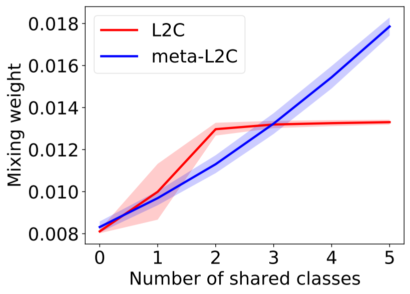


Figure 7. Mixing weights $w_{i,j}$ (mean±std over all (i, j) pairs) vs. the number of shared classes between node i and j on CIFAR-100. L2C and meta-L2C assign larger mixing weights to nodes sharing more classes and thus improves their local models' personalization performance.

In order to explain how the mixing weights produced by our methods can help training persoanlized models for non-IID tasks, we investigate the correlation between mixing weights and task similarities, e.g., the aggregation of models for similar tasks tend to improve the personalization performance for all of them. In Fig. 7, on the 100 nodes assigned with non-IID tasks from CIFAR-100, we observe that the mixing weight between two nodes grows as their shared classes increase, which demonstrates that L2C/meta-L2C encourage a node to collaborate with other nodes of similar tasks. To take a closer look at the tasks and mixing weights, we present a case study of two nodes and their neighbors in Fig. 6. It indicates that our methods can identify neighbors with not only shared classes but also semantically similar classes and assign larger mixing weights to them, hence improving the personalization performance of each node's local task.

4.3. Sensitivity Analysis

Train/val ratio For our methods and FOMO that uses a training set to update the local models and a validation set to determine the mixing weights, the train/val ratio controls their trade-off and its choice can be critical to the performance. In Fig. 8, we compare the sensitivity of different methods to this hyperparameter. While FOMO shows a sensitive trade-off, both of our methods are robust to the ratio

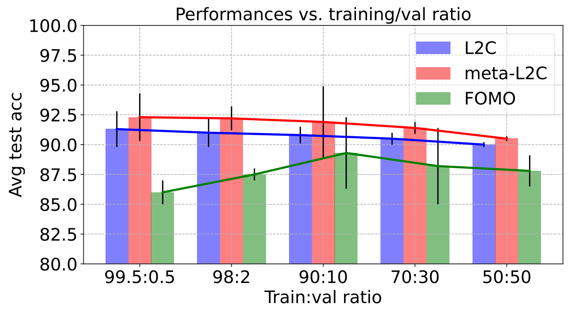


Figure 8. L2C and meta-L2C are more robust to the choices of train/val ratio than FOMO (the only baseline aiming at minimizing the validation loss).

in the evaluated interval, because L2C/meta-L2C only contains a few parameters and a small validation set suffices to estimate them.

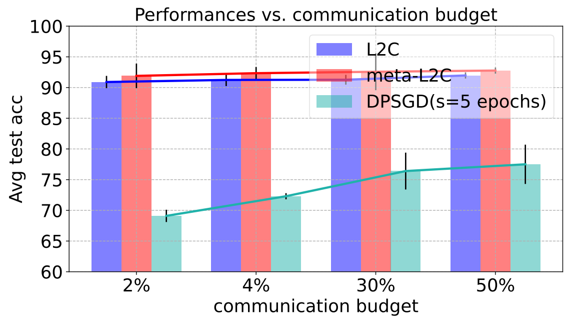


Figure 9. L2C and meta-L2C are more robust to different communication cost/budgets. We evaluate each method for every node communicating with 2%, 4%, 30%, or 50% of other nodes.

Communication cost/budget The communication cost can be a bottleneck of DL so a sparse topology is usually necessary in practice. Hence, in Fig. 9, we evaluate the personalization performance of L2C, meta-L2C, and DPSGD under different communication budgets. Both of our methods keep almost the same accuracy across the tested budgets, while DPSGD's performance is much poorer and sensitively improves only when the budget increases.

5. Conclusion

We propose "learning to collaborate (L2C)" and meta-L2C to automatically optimize the mixing weights in decentralized learning for producing better local models on non-IID tasks/data. By training a few parameters shared across tasks, our methods can precisely capture the correlation between nodes/tasks and mainly aggregate those most related to each local task. It also learns a sparse topology in earlier stages that can significantly reduce the communication cost. We demonstrate several practical advantages of our methods over a diverse set of recent FL/DL methods by experiments.

6. Acknowledgements

This work was supported by NSFC No. 61872329 and the Fundamental Research Funds for the Central Universities under contract WK3490000005.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021. 2

[2] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2022. 2

[3] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016. 3

[4] Hong-You Chen and Wei-Lun Chao. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021. 2

[5] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2089–2099. PMLR, 18–24 Jul 2021. 2, 3

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[7] Yasaman Esfandiari, Sin Yong Tan, Zhanhong Jiang, Aditya Balu, Ethan Herron, Chinmay Hegde, and Soumik Sarkar. Cross-gradient aggregation for decentralized learning from non-iid data. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3036–3046. PMLR, 18–24 Jul 2021. 2, 3, 5, 6

[8] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 3

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 3, 5

[10] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3407–3416. PMLR, 18–24 Jul 2021. 2

[11] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020. 2

[12] Andrew Gibiansky. Bringing hpc techniques to deep learning. *Baidu Research, Tech. Rep.*, 2017. 5

[13] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020. 2, 3, 5

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 5

[15] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. Collaborative deep learning in fixed topology networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[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. 2

[17] Sahar Khawatmi, Ali H Sayed, and Abdelhak M Zoubir. Decentralized clustering and linking by networked agents. *IEEE Transactions on Signal Processing*, 65(13):3526–3537, 2017. 3

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[20] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021. 2, 6

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018. 2

[22] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020. 2

[23] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. FedBN: Federated learning on non-IID features via local batch normalization. In *International Conference on Learning Representations*, 2021. 3

[24] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 3, 5, 6

[25] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and

Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020. 3, 5

[26] Tao Lin, Sai Praneeth Karimireddy, Sebastian Stich, and Martin Jaggi. Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6654–6665. PMLR, 18–24 Jul 2021. 3

[27] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2351–2363. Curran Associates, Inc., 2020. 2

[28] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning to propagate for graph meta-learning. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[29] 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*, pages 1273–1282. PMLR, 2017. 1, 2, 5, 6

[30] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. 5

[31] 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. 2

[32] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9489–9502. PMLR, 18–24 Jul 2021. 2, 3

[33] Canh T. Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21394–21405. Curran Associates, Inc., 2020. 2

[34] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI Conference on Artificial Intelligence*, volume 1, 2022. 3

[35] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020. 2

[36] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 5

[37] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021. 2

[38] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *Advances in Neural Information Processing Systems*, 34, 2021. 1

[39] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2021. 5, 6

[40] Qiming Zhang, Yufei Xu, Jing Zhang, and Dacheng Tao. Vitaev2: Vision transformer advanced by exploring inductive bias for image recognition and beyond. *arXiv preprint arXiv:2202.10108*, 2022. 1

[41] Tianyi Zhou and Dacheng Tao. Multi-task copula by sparse graph regression. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 771–780, 2014. 3

[42] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12878–12889. PMLR, 18–24 Jul 2021. 3