D-Cliques: Compensating for Data Heterogeneity with Topology in Decentralized Federated Learning

Aurélien Bellet, Anne-Marie Kermarrec, Erick Lavoie

D-CLIQUES: COMPENSATING FOR DATA HETEROGENEITY WITH TOPOLOGY IN DECENTRALIZED FEDERATED LEARNING

Aurélien Bellet [1] **Anne-Marie Kermarrec** [2] **Erick Lavoie** [2]

ABSTRACT

The convergence speed of machine learning models trained with Federated Learning is significantly affected by heterogeneous data partitions, even more so in a fully decentralized setting without a central server. In this paper, we show that the impact of label distribution skew, an important type of data heterogeneity, can be significantly reduced by carefully designing the underlying communication topology. We present D-Cliques, a novel topology that reduces gradient bias by grouping nodes in sparsely interconnected cliques such that the label distribution in a clique is representative of the global label distribution. We also show how to adapt the updates of decentralized SGD to obtain unbiased gradients and implement an effective momentum with D-Cliques. Our extensive empirical evaluation on MNIST and CIFAR10 demonstrates that our approach provides similar convergence speed as a fully-connected topology, which provides the best convergence in a data heterogeneous setting, with a significant reduction in the number of edges and messages. In a 1000-node topology, D-Cliques require 98% less edges and 96% less total messages, with further possible gains using a small-world topology across cliques.

1 INTRODUCTION

Machine learning is currently shifting from a *centralized* paradigm, where training data is located on a single machine or in a data center, to *decentralized* ones in which data is processed where it was naturally produced. This shift is illustrated by the rise of Federated Learning (FL) (McMahan et al., 2017). FL allows several parties (hospitals, companies, personal devices...) to collaboratively train machine learning models on their joint data without centralizing it. Not only does FL avoid the costs of moving data, but it also mitigates privacy and confidentiality concerns (Kairouz et al., 2021). Yet, working with natural data distributions introduces new challenges for learning systems, as local datasets reflect the usage and production patterns specific to each participant: in other words, they are *heterogeneous*. An important type of data heterogeneity encountered in federated classification problems, known as *label distribution skew* (Kairouz et al., 2021; Hsieh et al., 2020), occurs when the frequency of different classes of examples varies significantly across local datasets. One of the key challenges in FL is to design algorithms that can efficiently deal with such heterogeneous data distributions (Kairouz et al., 2021; Li et al., 2020; Karimireddy et al., 2020; Hsieh et al., 2020).

Federated learning algorithms can be classified into two categories depending on the underlying network topology they run on. In server-based FL, the network is organized according to a star topology: a central server orchestrates the training process by iteratively aggregating model updates received from the participants (*clients*) and sending back the aggregated model (McMahan et al., 2017). In contrast, fully decentralized FL algorithms operate over an arbitrary network topology where participants communicate only with their direct neighbors in the network. A classic example of such algorithms is Decentralized SGD (D-SGD) (Lian et al., 2017), in which participants alternate between local SGD updates and model averaging with neighboring nodes.

In this paper, we focus on fully decentralized algorithms as they can generally scale better to the large number of participants seen in "cross-device" applications (Kairouz et al., 2021). Effectively, while a central server may quickly become a bottleneck as the number of participants increases, the topology used in fully decentralized algorithms can remain sparse enough such that all participants need only to communicate with a small number of other participants, i.e. nodes have small (constant or logarithmic) degree (Lian et al., 2017). In the homogeneous setting where data is independent and identically distributed (IID) across nodes, recent work has shown both empirically (Lian et al., 2017; 2018) and theoretically (Neglia et al., 2020) that sparse topologies like rings or grids do not significantly affect the convergence speed compared to using denser topologies.

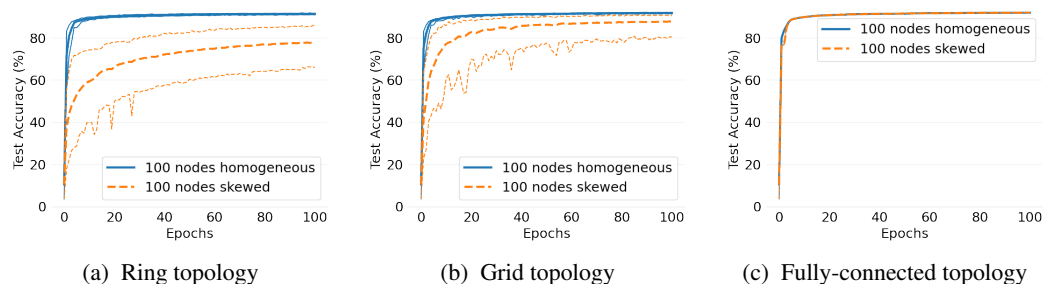In contrast to the homogeneous case however, our experi-

---

[1]Inria, Lille, France [2]EPFL, Lausanne, Switzerland. Correspondence to: Erick Lavoie <erick.lavoie@epfl.ch>.

Figure 1. Convergence speed of decentralized SGD with and without label distribution skew for different topologies. The task is logistic regression on MNIST (see Section 4.1 for details on the experimental setup). Bold lines show the average test accuracy across nodes while thin lines show the minimum and maximum accuracy of individual nodes. While the effect of topology is negligible for homogeneous data, it is very significant in the heterogeneous case. On a fully-connected network, both cases converge similarly.

ments demonstrate that *the impact of topology is extremely significant for heterogeneous data*. This phenomenon is illustrated in Figure 1: we observe that under label distribution skew, using a sparse topology (a ring or a grid) clearly jeopardizes the convergence speed of decentralized SGD. We stress the fact that, unlike in centralized FL (McMahan et al., 2017; Karimireddy et al., 2020; Hsieh et al., 2020), this happens even when nodes perform a single local update before averaging the model with their neighbors. In this paper, we thus address the following question:

Can we design sparse topologies with convergence speed similar to a fully connected network for problems involving many participants with label distribution skew?

Specifically, we make the following contributions: (1) We propose D-Cliques, a sparse topology in which nodes are organized in interconnected cliques (i.e., locally fully-connected sets of nodes) such that the joint label distribution of each clique is close to that of the global distribution; (2) We design Greedy Swap, a randomized greedy algorithm for constructing such cliques efficiently; (3) We introduce Clique Averaging, a modified version of the standard D-SGD algorithm which decouples gradient averaging, used for optimizing local models, from distributed averaging, used to ensure that all models converge, thereby reducing the bias introduced by inter-clique connections; (4) We show how Clique Averaging can be used to implement unbiased momentum that would otherwise be detrimental in the heterogeneous setting; (5) We demonstrate through an extensive experimental study that our approach removes the effect of label distribution skew when training a linear model and a deep convolutional network on the MNIST and CIFAR10 datasets respectively; (6) Finally, we demonstrate the scalability of our approach by considering up to 1000-node networks, in contrast to most previous work on fully decentralized learning which performs empirical evaluations on networks with at most a few tens of nodes (Tang et al., 2018; Neglia et al., 2020; Lin et al., 2021; Esfandiari et al., 2021; Kong et al., 2021).

For instance, our results show that under strong label distribution shift, using D-Cliques in a 1000-node network requires 98% less edges (18.9 vs 999 edges per participant on average) to obtain a similar convergence speed as a fully-connected topology, thereby yielding a 96% reduction in the total number of required messages (37.8 messages per round per node on average instead of 999). Furthermore an additional 22% improvement is possible when using a small-world inter-clique topology, with further potential gains at larger scales through a quasilinear $O(n \log n)$ scaling in the number of nodes n.

The rest of this paper is organized as follows. We first describe the problem setting in Section 2. We then present the design of D-Cliques in Section 3. Section 4 compares D-Cliques to different topologies and algorithmic variations to demonstrate their benefits, constructed with and without Greedy Swap in an extensive experimental study. Finally, we review some related work in Section 5, and conclude with promising directions for future work in Section 6.

2 PROBLEM SETTING

Objective. We consider a set $N = \{1, \ldots, n\}$ of n nodes seeking to collaboratively solve a classification task with L classes. We denote a labeled data point by a tuple (x, y) where x represents the data point (e.g., a feature vector) and $y \in \{1, \ldots, L\}$ its label. Each node has access to a local dataset that follows its own local distribution D_i which may differ from that of other nodes. In this work, we tackle *label distribution skew*: formally, this means that the probability of (x, y) under the local distribution D_i of node i, denoted by $p_i(x, y)$, decomposes as $p_i(x, y) = p(x|y)p_i(y)$, where $p_i(y)$ may vary across nodes. We refer to (Kairouz et al., 2021; Hsieh et al., 2020) for concrete examples of problems with label distribution skew.

The objective is to find the parameters θ of a global model that performs well on the union of the local distributions by

Algorithm 1 D-SGD, Node i

1: **Require:** initial model $\theta_i^{(0)}$, learning rate γ, mixing weights W, mini-batch size m, number of steps K
2: **for** $k = 1, \ldots, K$ **do**
3:     $S_i^{(k)} \leftarrow$ mini-batch of m samples drawn from D_i
4:     $\theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma \nabla F(\theta_i^{(k-1)}; S_i^{(k)})$
5:     $\theta_i^{(k)} \leftarrow \sum_{j \in N} W_{ji}^{(k)} \theta_j^{(k-\frac{1}{2})}$

(a) Homogeneous data   (b) Heterogeneous data

Figure 2. Neighborhood in a grid.

minimizing the average training loss:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{(x_i, y_i) \sim D_i}[F_i(\theta; x_i, y_i)], \qquad (1)$$

where (x_i, y_i) is a data point drawn from D_i and F_i is the loss function on node i. Therefore, $\mathbb{E}_{(x_i, y_i) \sim D_i} F_i(\theta; x_i, y_i)$ denotes the expected loss of model θ over D_i.

To collaboratively solve Problem (1), each node can exchange messages with its neighbors in an undirected network graph $G = (N, E)$ where $\{i, j\} \in E$ denotes an edge (communication channel) between nodes i and j.

Training algorithm. In this work, we use the popular Decentralized Stochastic Gradient Descent algorithm, aka D-SGD (Lian et al., 2017). As shown in Algorithm 1, a single iteration of D-SGD at node i consists in sampling a mini-batch from its local distribution D_i, updating its local model θ_i by taking a stochastic gradient descent (SGD) step according to the mini-batch, and performing a weighted average of its local model with those of its neighbors. This weighted average is defined by a mixing matrix W, in which W_{ij} corresponds to the weight of the outgoing connection from node i to j and $W_{ij} = 0$ for $\{i, j\} \notin E$. To ensure that the local models converge on average to a stationary point of Problem (1), W must be doubly stochastic ($\sum_{j \in N} W_{ij} = 1$ and $\sum_{j \in N} W_{ji} = 1$) and symmetric, i.e. $W_{ij} = W_{ji}$ (Lian et al., 2017). Given a network topology $G = (N, E)$, we generate a valid W by computing standard Metropolis-Hasting weights (Xiao & Boyd, 2004):

$$W_{ij} = \begin{cases} \frac{1}{\max(\text{degree}(i), \text{degree}(j)) + 1} & \text{if } i \neq j \text{ and } \{i, j\} \in E, \\ 1 - \sum_{j \neq i} W_{ij} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

$$(2)$$

3 D-CLIQUES

In this section, we introduce D-Cliques, a topology designed to compensate for data heterogeneity. We also present some modifications of D-SGD that leverage some properties of the proposed topology and allow to implement a successful momentum scheme.
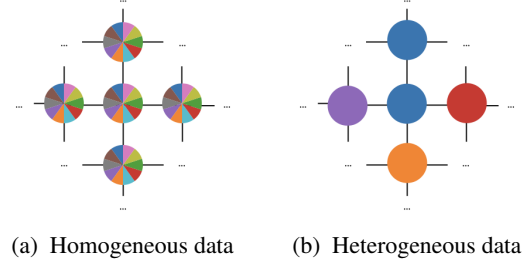
3.1 Intuition

To give the intuition behind our approach, let us consider the neighborhood of a single node in a grid topology represented on Figure 2. Nodes are distributed randomly in the grid and the colors of a node represent the proportion of each class in its local dataset. In the homogeneous setting, the label distribution is the same across nodes: in the example shown in Figure 2a, all classes are represented in equal proportions on all nodes. This is not the case in the heterogeneous setting: Figure 2b shows an extreme case of label distribution skew where each node holds examples of a single class only.

From the point of view of the center node in Figure 2, a single training step of D-SGD is equivalent to sampling a mini-batch five times larger from the union of the local distributions of neighboring nodes. In the homogeneous case, since gradients are computed from examples of all classes, the resulting averaged gradient points in a direction that tends to reduce the loss across all classes. In contrast, in the heterogeneous case, the representation of classes in the immediate neighborhood of the node is different from the global label distribution (in Figure 2b, only a subset of classes are represented), thus the gradients will be biased. Importantly, as the distributed averaging process takes several steps to converge, this variance persists across iterations as the locally computed gradients are far from the global average.[1] This can significantly slow down convergence speed to the point of making decentralized optimization impractical.

With D-Cliques, we address label distribution skew by carefully designing a network topology composed of *locally representative cliques* while maintaining *sparse inter-clique connections* only.

3.2 Constructing Locally Representative Cliques

D-Cliques construct a topology in which each node is part of a *clique* (i.e., a subset of nodes whose induced subgraph

---

[1] One could perform a sufficiently large number of averaging steps between each gradient step, but this is too costly in practice.

Algorithm 2 D-Cliques Construction via Greedy Swap

1: **Require:** maximum clique size M, max steps K, set of all nodes $N = \{1, 2, \ldots, n\}$, procedure $\texttt{inter}(\cdot)$ to create intra-clique connections (see Sec. 3.3)
2: $DC \leftarrow []$
3: **while** $N \neq \emptyset$ **do**
4: $\quad C \leftarrow$ sample M nodes from N at random
5: $\quad N \leftarrow N \setminus C$; $DC.\text{append}(C)$
6: **for** $k \in \{1, \ldots, K\}$ **do**
7: $\quad C_1, C_2 \leftarrow$ random sample of 2 elements from DC
8: $\quad s \leftarrow skew(C_1) + skew(C_2)$
9: $\quad swaps \leftarrow []$
10: $\quad$ **for** $i \in C_1, j \in C_2$ **do**
11: $\quad\quad s' \leftarrow skew(C_1 \setminus \{i\} \cup \{j\}) + skew(C_2 \setminus \{i\} \cup \{j\})$
12: $\quad\quad$ **if** $s' < s$ **then**
13: $\quad\quad\quad swaps.\text{append}((i, j))$
14: $\quad$ **if** $len(swaps) > 0$ **then**
15: $\quad\quad (i, j) \leftarrow$ random element from $swaps$
16: $\quad\quad C_1 \leftarrow C_1 \setminus \{i\} \cup \{j\}$; $C_2 \leftarrow C_2 \setminus \{j\} \cup \{i\}$
17: $E \leftarrow \{(i, j) : C \in DC, i, j \in C, i \neq j\}$
18: **return** topology $G = (N, E \cup \texttt{inter}(DC))$

is fully connected) such that the label distribution in each clique is close to the global label distribution. Formally, for a label y and a clique composed of nodes $C \subseteq N$, we denote by $p_C(y) = \frac{1}{|C|} \sum_{i \in C} p_i(y)$ the distribution of y in C and by $p(y) = \frac{1}{n} \sum_{i \in N} p_i(y)$ its global distribution. We measure the *skew* of C by the sum of the absolute differences of $p_C(y)$ and $p(y)$:

$$skew(C) = \sum_{l=1}^{L} |p_C(y = l) - p(y = l)|. \quad (3)$$

To efficiently construct a set of cliques with small skew, we propose Greedy-Swap (Algorithm 2). The parameter M is the maximum size of cliques and controls the number of intra-clique edges. We start by initializing cliques at random. Then, for a certain number of steps K, we randomly pick two cliques and swap two of their nodes so as to decrease the sum of skews of the two cliques. The swap is chosen randomly among the ones that decrease the skew, hence this algorithm can be seen as a form of randomized greedy algorithm. We note that this algorithm only requires the knowledge of the label distribution $p_i(y)$ at each node i. For the sake of simplicity, we assume that D-Cliques are constructed from the global knowledge of these distributions, which can easily be obtained by decentralized averaging in a pre-processing step (e.g., Jelasity et al., 2005).

The key idea of D-Cliques is to ensure the clique-level label distribution $p_C(y)$ matches closely the global distribution $p(y)$. As a consequence, the local models of nodes across cliques remain rather close. Therefore, a sparse inter-clique

topology can be used, significantly reducing the total number of edges without slowing down the convergence. We discuss some possible choices for this inter-clique topology in the next section.

3.3 Adding Sparse Inter-Clique Connections

To ensure a global consensus and convergence, we introduce *inter-clique connections* between a small number of node pairs that belong to different cliques, thereby implementing the $\texttt{inter}$ procedure called at the end of Algorithm 2. We aim to ensure that the degree of each node remains low and balanced so as to make the network topology well-suited to decentralized federated learning. We consider several choices of inter-clique topology, which offer different scalings for the number of required edges and the average distance between nodes in the resulting graph.

The *ring* has (almost) the fewest possible number of edges for the graph to be connected: in this case, each clique is connected to exactly two other cliques by a single edge. This topology requires only $O(\frac{n}{M})$ inter-clique edges but suffers an $O(n)$ average distance between nodes.

The *fractal* topology provides a logarithmic bound on the average distance. In this hierarchical scheme, cliques are arranged in larger groups of M cliques that are connected internally with one edge per pair of cliques, but with only one edge between pairs of larger groups. The topology is built recursively such that M groups will themselves form a larger group at the next level up. This results in at most M edges per node if edges are evenly distributed: i.e., each group within the same level adds at most $M - 1$ edges to other groups, leaving one node per group with $M - 1$ edges that can receive an additional edge to connect with other groups at the next level. Since nodes have at most M edges, the total number of inter-clique edges is at most nM edges.

We can also design an inter-clique topology in which the number of edges scales in a log-linear fashion by following a small-world-like topology (Watts, 2000) applied on top of a ring (Stoica et al., 2003). In this scheme, cliques are first arranged in a ring. Then each clique adds symmetric edges, both clockwise and counter-clockwise on the ring, with the c closest cliques in sets of cliques that are exponentially bigger the further they are on the ring (see Algorithm 4 in Appendix A for details on the construction). This topology ensures a good connectivity with other cliques that are close on the ring, while keeping the average distance small. This scheme uses $O(c\frac{n}{M} \log \frac{n}{M})$ edges, i.e. log-linear in n.

Finally, we can consider a *fully connected* inter-clique topology such that each clique has exactly one edge with each of the other cliques, spreading these additional edges equally among the nodes of a clique, as illustrated in Figure 3. This has the advantage of bounding the distance
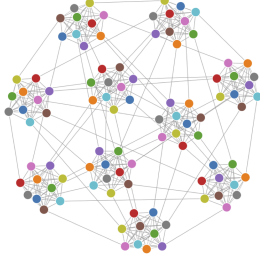
Figure 3. D-Cliques with $n = 100$, $M = 10$ and a fully connected inter-clique topology on a problem with 1 class/node.
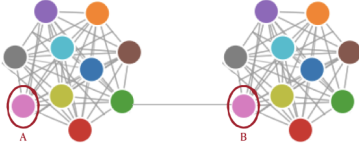

Algorithm 3 D-SGD with Clique Averaging, Node i

1: **Require** initial model $\theta_i^{(0)}$, learning rate γ, mixing weights W, mini-batch size m, number of steps K
2: **for** $k = 1, \ldots, K$ **do**
3: $\quad S_i^{(k)} \leftarrow$ mini-batch of m samples drawn from D_i
4: $\quad g_i^{(k)} \leftarrow \frac{1}{|Clique(i)|} \sum_{j \in Clique(i)} \nabla F(\theta_j^{(k-1)}; S_j^{(k)})$
5: $\quad \theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma g_i^{(k)}$
6: $\quad \theta_i^{(k)} \leftarrow \sum_{j \in N} W_{ji}^{(k)} \theta_j^{(k-\frac{1}{2})}$

Figure 4. Illustrating the bias induced by inter-clique connections (see main text for details).

between any pair of nodes to 3 but requires $O(\frac{n^2}{M^2})$ inter-clique edges, i.e. quadratic in n.

3.4 Optimizing over D-Cliques with Clique Averaging and Momentum

While limiting the number of inter-clique connections reduces the amount of messages traveling on the network, it also introduces a form of bias. Figure 4 illustrates the problem on the simple case of two cliques connected by one inter-clique edge (here, between the green node of the left clique and the pink node of the right clique). In this example, each node holds example of a single class. Let us focus on node A. With weights computed as in (2), node A's self-weight is $\frac{12}{110}$, the weight between A and the green node connected to B is $\frac{10}{110}$, and all other neighbors of A have a weight of $\frac{11}{110}$. Therefore, the gradient at A is biased towards its own class (pink) and against the green class. A similar bias holds for all other nodes without inter-clique edges with respect to their respective classes. For node B, all its edge weights (including its self-weight) are equal to $\frac{1}{11}$. However, the green class is represented twice (once as a clique neighbor and once from the inter-clique edge), while all other classes are represented only once. This biases the gradient toward the green class. The combined effect of these two sources of bias is to increase the variance of the local models across nodes.

Clique Averaging. We address this problem by adding *Clique Averaging* to D-SGD (Algorithm 3), which essentially decouples gradient averaging from model averaging. The idea is to use only the gradients of neighbors within

the same clique to compute the average gradient so as to remove the bias due to inter-clique edges. In contrast, all neighbors' models (including those in different cliques) participate in model averaging as in the original version. Adding Clique Averaging requires gradients to be sent separately from the model parameters: the number of messages exchanged between nodes is therefore twice their number of edges.

Implementing momentum with Clique Averaging. Efficiently training high capacity models usually requires additional optimization techniques. In particular, momentum (Sutskever et al., 2013) increases the magnitude of the components of the gradient that are shared between several consecutive steps, and is critical for deep convolutional networks like LeNet (LeCun et al., 1998; Hsieh et al., 2020) to converge quickly. However, a direct application of momentum in data heterogeneous settings can actually be very detrimental and even fail to converge, as we will show in our experiments (Figure 7 in Section 4). Clique Averaging allows us to reduce the bias in the momentum by using the clique-level average gradient $g_i^{(k)}$ of Algorithm 3:

$$v_i^{(k)} \leftarrow m v_i^{(k-1)} + g_i^{(k)}. \tag{4}$$

It then suffices to modify the original gradient step to apply momentum:

$$\theta_i^{(k-\frac{1}{2})} \leftarrow \theta_i^{(k-1)} - \gamma v_i^{(k)}. \tag{5}$$

4 EVALUATION

In this section, we first compare D-Cliques to alternative topologies to show the benefits and relevance of our main design choices. Then, we evaluate different inter-clique topologies to further reduce the number of inter-clique connections so as to gracefully scale with the number of nodes. Then, we show the impact of removing intra-clique edges. Finally, we show that Greedy Swap (Alg. 2) constructs cliques efficiently with consistently lower skew than random cliques.

4.1 Experimental Setup

Our main goal is to provide a fair comparison of the convergence speed across different topologies and algorithmic variations, in order to show that D-Cliques can remove much of the effects of label distribution skew.

We experiment with two datasets: MNIST (LeCun et al., 2020) and CIFAR10 (Krizhevsky, 2009), which both have $L = 10$ classes. For MNIST, we use 50k and 10k examples from the original 60k training set for training and validation respectively. We use all 10k examples of the test set to measure prediction accuracy. The validation set preserves the original unbalanced ratio of the classes in the test set, and the remaining examples become the training set. For CIFAR10, classes are evenly balanced: we initially used 45k/50k images of the original training set for training, 5k/50k for validation, and all 10k examples of the test set for measuring prediction accuracy. After tuning hyperparameters on initial experiments, we then used all 50k images of the original training set for training for all experiments, as the 45k did not split evenly in 1000 nodes with the partitioning scheme explained in the next paragraph.

For both MNIST and CIFAR10, we use the heterogeneous data partitioning scheme proposed by McMahan et al. (2017) in their seminal FL work: we sort all training examples by class, then split the list into shards of equal size, and randomly assign two shards to each node. When the number of examples of one class does not divide evenly in shards, as is the case for MNIST, some shards may have examples of more than one class and therefore nodes may have examples of up to 4 classes. However, most nodes will have examples of 2 classes. The varying number of classes, as well as the varying distribution of examples within a single node, makes the task of creating cliques with low skew nontrivial.

We use a logistic regression classifier for MNIST, which provides up to 92.5% accuracy in the centralized setting. For CIFAR10, we use a Group-Normalized variant of LeNet (Hsieh et al., 2020), a deep convolutional network which achieves an accuracy of 74.15% in the centralized setting. These models are thus reasonably accurate (which is sufficient to study the effect of the topology) while being sufficiently fast to train in a fully decentralized setting and simple enough to configure and analyze. Regarding hyperparameters, we jointly optimize the learning rate and minibatch size on the validation set for 100 nodes, obtaining respectively 0.1 and 128 for MNIST and 0.002 and 20 for CIFAR10. For CIFAR10, we additionally use a momentum of 0.9.

We evaluate 100- and 1000-node networks by creating multiple models in memory and simulating the exchange of messages between nodes. To ignore the impact of distributed execution strategies and system optimization techniques, we report the test accuracy of all nodes (min, max, average) as a function of the number of times each example of the dataset has been sampled by a node, i.e. an *epoch*. This is equivalent to the classic case of a single node sampling the full distribution. To further make results comparable across different number of nodes, we lower the batch size proportionally to the number of nodes added, and inversely, e.g. on MNIST, 128 with 100 nodes vs. 13 with 1000 nodes. This ensures the same number of model updates and averaging per epoch, which is important to have a fair comparison.[2]

Finally, we compare our results against an ideal baseline: a fully-connected network topology with the same number of nodes. This baseline is essentially equivalent to a centralized (single) IID node using a batch size n times bigger, where n is the number of nodes. Both a fully-connected network and a single IID node effectively optimize a single model and sample uniformly from the global distribution: both therefore remove entirely the effect of label distribution skew and of the network topology on the optimization. In practice, we prefer a fully-connected network because it converges slightly faster and obtains slightly better final accuracy than a single node sampling randomly from the global distribution.[3]

4.2 D-Cliques Match the Convergence Speed of Fully-Connected with a Fraction of the Edges

In this first experiment, we show that D-Cliques with Clique Averaging (and momentum when mentioned) converges almost as fast as a fully-connected network on both MNIST and CIFAR10. Figure 5 illustrates the convergence speed of D-Cliques with $n = 100$ nodes on MNIST (with Clique Averaging) and CIFAR10 (with Clique Averaging and momentum). Observe that the convergence speed is very close to that of a fully-connected topology, and significantly better than with a ring or a grid (see Figure 1). It also has less variance than both the ring and grid.

4.3 Clique Averaging is Beneficial and Sometimes Necessary

In this experiment, we perform an ablation study of the effect of Clique Averaging. Figure 6 shows that Clique Averaging (Algorithm 3) reduces the variance of models across nodes and slightly accelerates the convergence on

---

[2]Updating and averaging models after every example can eliminate the impact of label distribution skew. However, the resulting communication overhead is impractical.

[3]We conjecture that an heterogeneous data partition in a fully-connected network may force more balanced representation of all classes in the union of all mini-batches, leading to better convergence.
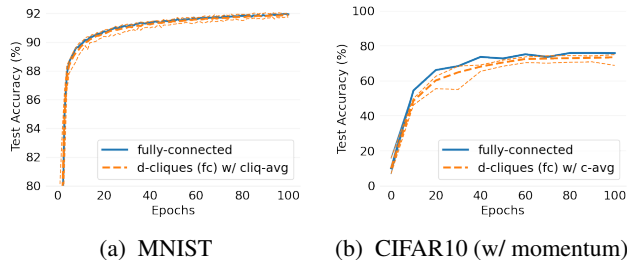
(a) MNIST (b) CIFAR10 (w/ momentum)

Figure 5. Comparison on 100 heterogeneous nodes (2 shards/node) between a fully-connected network and D-Cliques (fully-connected) constructed with Greedy Swap (10 cliques of 10 nodes) using Clique Averaging. Bold line is the average accuracy over all nodes. Thinner upper and lower lines are maximum and minimum accuracy over all nodes.

MNIST. Recall that Clique Averaging induces a small additional cost, as gradients and models need to be sent in two separate rounds of messages. Nonetheless, compared to fully connecting all nodes, the total number of messages per round for 100 nodes is reduced by $\approx 80\%$.
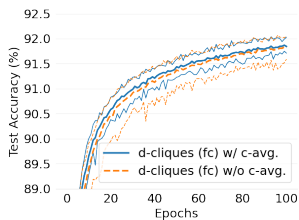
Figure 6. MNIST: Effect of Clique Averaging on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes). Y axis starts at 89.

The effect of Clique Averaging is much more pronounced on CIFAR10, as can be seen in Figure 7, especially when used in combination with momentum. Without Clique Averaging, the use of momentum is actually detrimental. With Clique Averaging, the situation reverses and momentum is again beneficial. The combination of both has the fastest convergence speed and the lowest variance among all four possibilities. We believe that the gains obtained with Clique Averaging are larger on CIFAR10 than on MNIST because the model we train on CIFAR10 (a deep convolutional network) has much higher capacity than the linear model used for MNIST. The resulting highly non-convex objective increases the sensitivity of local updates to small differences in the gradients, making them point in different directions, as observed by Kong et al. (2021) even in the homogeneous setting. Clique Averaging helps to reduce this effect by reducing the bias in local gradients.
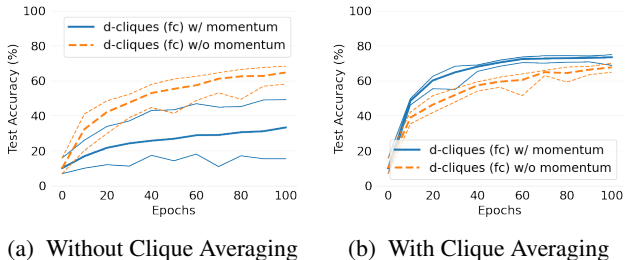

(a) Without Clique Averaging (b) With Clique Averaging

Figure 7. CIFAR10: Effect of Clique Averaging, without and with momentum, on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes).

4.4 D-Cliques Converge Faster than Random Graphs

In this experiment, we compare D-Cliques to a random graph that has a similar number of edges (10) per node to determine whether a simple sparse topology could work equally well. To ensure a fair comparison, because a random graph does not support Clique Averaging, we do not use it for D-Cliques either. Figure 8 shows that even *without* Clique Averaging, D-Cliques converge faster and with lower variance. Furthermore, the use of momentum in a random graph is detrimental, similar to D-Cliques without the use of Clique Averaging (see Figure 7a). This shows that a careful design of the topology is indeed necessary.

D-Cliques converge faster even if we were to create diverse neighborhoods in a random graph with lower skew and used those to unbias gradients in an analogous way to Clique Averaging (details in Annex C.3.4, as the experiments require a different partitioning scheme for a fair comparison). The clustering provided by D-Cliques therefore provides faster convergence.
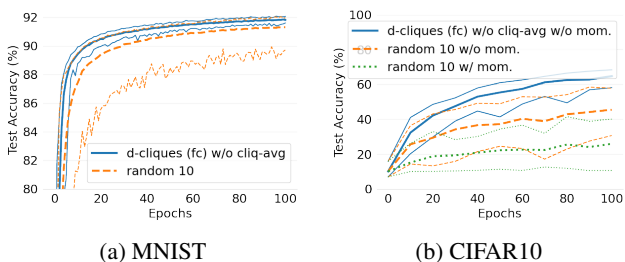

(a) MNIST (b) CIFAR10

Figure 8. Comparison on 100 heterogeneous nodes between D-Cliques (fully-connected) with 10 cliques of size 10 and a random graph with 10 edges per node *without* Clique Averaging or momentum.

4.5 D-Cliques Scale with Sparser Inter-Clique Topologies

In this experiment, we explore the trade-offs between scalability and convergence speed induced by the several sparse inter-clique topologies introduced in Section 3.3. Figure 9

and Figure 10 show the convergence speed respectively on MNIST and CIFAR10 on a larger network of 1000 nodes, compared to the ideal baseline of a fully-connected network representing the fastest convergence speed achievable if topology had no impact. Among the linear schemes, the ring topology converges but is much slower than our fractal scheme. Among the super-linear schemes, the small-world topology has a convergence speed that is almost the same as with a fully-connected inter-clique topology but with 22% less edges (14.5 edges on average instead of 18.9).

(a) Linear   (b) Super- and Quasi-Linear
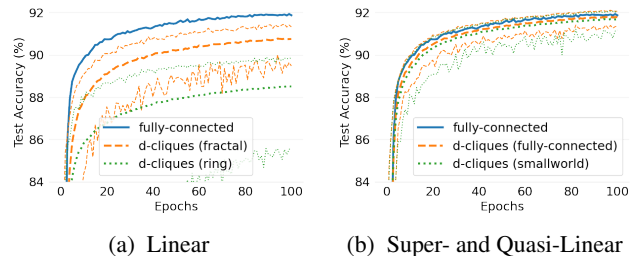
Figure 9. MNIST: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) and different inter-clique topologies.
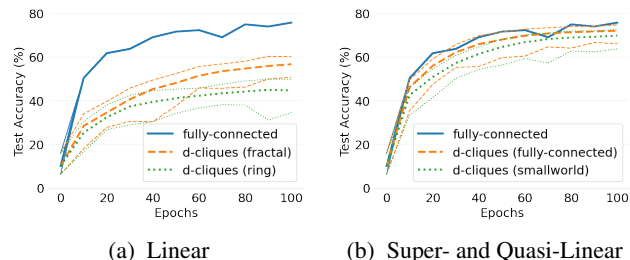

(a) Linear   (b) Super- and Quasi-Linear

Figure 10. CIFAR10: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) and different inter-clique topologies.

While the small-world inter-clique topology shows promising scaling behavior, the fully-connected inter-clique topology still offers significant benefits with 1000 nodes, as it represents a 98% reduction in the number of edges compared to fully connecting individual nodes (18.9 edges on average instead of 999) and a 96% reduction in the number of messages (37.8 messages per round per node on average instead of 999). We refer to Appendix B for additional results comparing the convergence speed across different number of nodes. Overall, these results show that D-Cliques can gracefully scale with the number of nodes.

4.6 Full Intra-Clique Connectivity is Necessary

In this experiment, we measure the impact of removing intra-clique edges to assess how critical full connectivity

is within cliques. We choose edges to remove among the 45 undirected edges present in cliques of size 10. The removal of an edge removes the connection in both directions. We remove 1 and 5 edges randomly, respectively 2.2% and 11% of intra-clique edges. Figure 11 shows that for MNIST, when not using Clique Averaging, removing edges decreases slightly the convergence speed and increases the variance between nodes. When using Clique Averaging, removing up to 5 edges does not noticeably affect the convergence speed and variance.
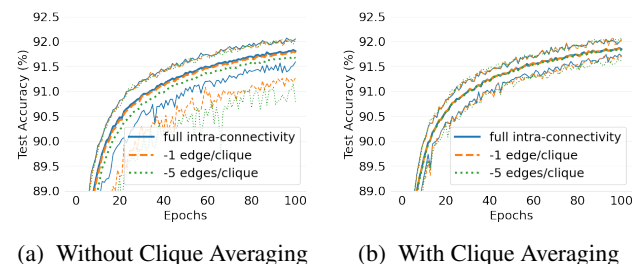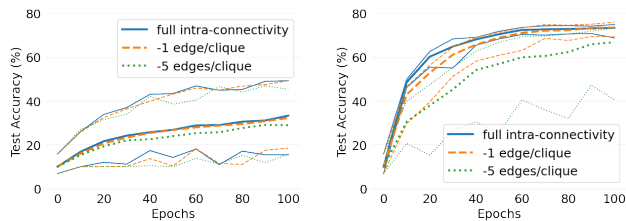

(a) Without Clique Averaging   (b) With Clique Averaging

Figure 11. MNIST: Impact of intra-clique edge removal on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes). Y axis starts at 89.

In contrast, Figure 12 shows that for CIFAR10, the impact is stronger. We show the results with and without Clique Averaging with momentum in both cases, as momentum is critical for obtaining the best convergence speed on CIFAR10. Without Clique Averaging, removing edges has a small effect on convergence speed and variance, but the convergence speed is too slow to be practical. With Clique Averaging, removing a single edge has a small but noticeable effect. Strikingly, removing 5 edges per clique significantly damages the convergence and yields a sharp increase in the variance across nodes. Therefore, while D-Cliques can tolerate the removal of some intra-clique edges when training simple linear models and datasets as in MNIST, fast convergence speed and low variance requires full or nearly full connectivity when using high-capacity models and more difficult datasets. This is in line with the observations made in Section 4.3 regarding the effect of Clique Averaging. Again, these results show the relevance of our design choices, including the choice of constructing fully connected cliques.

4.7 Greedy Swap Improves Random Cliques at an Affordable Cost

In the next two sub-sections, we compare cliques built with Greedy Swap (Alg. 2) to Random Cliques, a simple and obvious baseline, on their quality (skew), the cost of their construction, and their convergence speed.

(a) Without Clique Averaging   (b) With Clique Averaging

Figure 12. CIFAR10: Impact of intra-clique edge removal (with momentum) on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes).

4.7.1 Cliques with Low Skew can be Constructed Efficiently with Greedy Swap

We compared the final average skew of 10 cliques with 10 nodes each (for $n = 100$) created either randomly or with Greedy Swap, over 100 experiments after 1000 steps. Figure 14, in the form of an histogram, shows that Greedy Swap generates cliques of significantly lower skew, close to 0 in a majority of cases for both MNIST and CIFAR10.
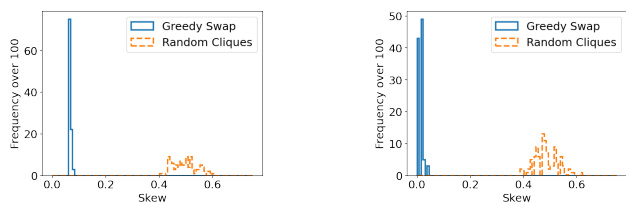
(a) MNIST   (b) CIFAR10

Figure 13. Final quality of cliques (skew) with a maximum size of 10 over 100 experiments in a network of 100 nodes.

Figure 14 shows such a low skew can be achieved in less than 400 steps for both MNIST and CIFAR10. In practice it takes less than 6 seconds in Python 3.7 on a Macbook Pro 2020 for a network of 100 nodes and cliques of size 10. Greedy Swap is therefore fast and efficient. Moreover, it illustrates the fact that a global imbalance in the number of examples across classes makes the construction of cliques with low skew harder and slower.

4.7.2 Cliques built with Greedy Swap Converge Faster than Random Cliques

Figure 15 compares the convergence speed of cliques optimized with Greedy Swap for 1000 steps with cliques built randomly (equivalent to Greedy Swap with 0 steps). For both MNIST and CIFAR10, convergence speed increases significantly and variance between nodes decreases dramatically. Decreasing the skew of cliques is therefore critical to convergence speed.
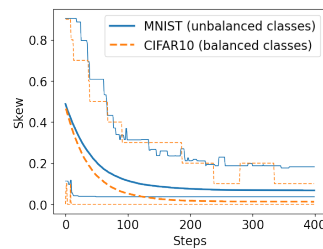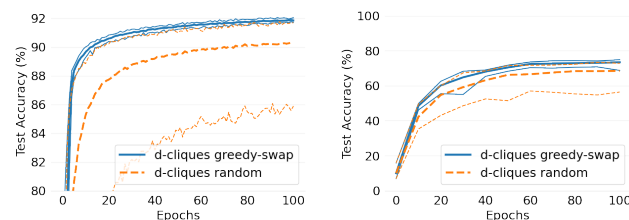

Figure 14. Skew decrease during clique construction of 10 cliques of 10 heterogeneous nodes (100 nodes). Bold line is the average over 100 experiments. Thin lines are respectively the minimum and maximum over all experiments. In wall-clock time, 1000 steps take less than 6 seconds in Python 3.7 on a MacBook Pro 2020.

(a) MNIST   (b) CIFAR10

Figure 15. Convergence speed of D-Cliques constructed randomly vs Greedy Swap with 10 cliques of 10 heterogeneous nodes (100 nodes).

4.8 Additional Experiments on Extreme Label Distribution Skew

In Appendix C, we replicate experimental results on an extreme case of label distribution skew where each node only has examples of a single class. These results consistently show that our approach remains effective even for extremely skewed label distributions across nodes.

5 RELATED WORK

In this section, we review some related work on dealing with heterogeneous data in federated learning, and on the role of topology in fully decentralized algorithms.

Dealing with heterogeneity in server-based FL. Data heterogeneity is not much of an issue in server-based FL if clients send their parameters to the server after each gradient update. Problems arise when one seeks to reduce the number of communication rounds by allowing each participant to perform multiple local updates, as in the popular FedAvg algorithm (McMahan et al., 2017). Indeed, data heterogeneity can prevent such algorithms from converging to a good solution (Hsieh et al., 2020; Karimireddy et al., 2020). This led to the design of algorithms that are specifically designed to mitigate the impact of heterogene-

ity while performing multiple local updates, using adaptive client sampling (Hsieh et al., 2020), update corrections (Karimireddy et al., 2020) or regularization in the local objective (Li et al., 2020). Another direction is to embrace the heterogeneity by learning personalized models for each client (Smith et al., 2017; Hanzely et al., 2020; Fallah et al., 2020; Dinh et al., 2020; Marfoq et al., 2021). We note that recent work explores rings of server-based topologies (Lee et al., 2020), but the focus is not on dealing with heterogeneous data but to make server-based FL more scalable to a large number of clients.

Dealing with heterogeneity in fully decentralized FL.
Data heterogeneity is known to negatively impact the convergence speed of fully decentralized FL algorithms in practice (Hegedüs et al., 2021). Aside from approaches that aim to learn personalized models (Vanhaesebrouck et al., 2017; Zantedeschi et al., 2020), this motivated the design of algorithms with modified updates based on variance reduction (Tang et al., 2018), momentum correction (Lin et al., 2021), cross-gradient aggregation (Esfandiari et al., 2021), or multiple averaging steps between updates (see Kong et al., 2021, and references therein). These algorithms typically require significantly more communication and/or computation, and have only been evaluated on small-scale networks with a few tens of nodes.[4] In contrast, D-Cliques focuses on the design of a sparse topology which is able to compensate for the effect of heterogeneous data and scales to large networks. We do not modify the simple and efficient D-SGD algorithm (Lian et al., 2017) beyond removing some neighbor contributions that otherwise bias the gradient direction.

Impact of topology in fully decentralized FL. It is well known that the choice of network topology can affect the convergence of fully decentralized algorithms. In theoretical convergence rates, this is typically accounted for by a dependence on the spectral gap of the network, see for instance (Duchi et al., 2012; Colin et al., 2016; Lian et al., 2017; Nedić et al., 2018). However, for homogeneous (IID) data, practice contradicts these classic results as fully decentralized algorithms have been observed to converge essentially as fast on sparse topologies like rings or grids as they do on a fully connected network (Lian et al., 2017; 2018). Recent work (Neglia et al., 2020; Kong et al., 2021) sheds light on this phenomenon with refined convergence analyses based on differences between gradients or parameters across nodes, which are typically smaller in the homogeneous case. However, these results do not give any

clear insight regarding the role of the topology in the presence of heterogeneous data. We note that some work has gone into designing efficient topologies to optimize the use of network resources (see e.g., Marfoq et al., 2020), but the topology is chosen independently of how data is distributed across nodes. In summary, the role of topology in the heterogeneous data scenario is not well understood and we are not aware of prior work focusing on this question. Our work is the first to show that an appropriate choice of data-dependent topology can effectively compensate for heterogeneous data.

6 CONCLUSION

We proposed D-Cliques, a sparse topology that obtains similar convergence speed as a fully-connected network in the presence of label distribution skew. D-Cliques is based on assembling subsets of nodes into cliques such that the clique-level class distribution is representative of the global distribution, thereby locally recovering homogeneity of data. Cliques are connected together by a sparse inter-clique topology so that they quickly converge to the same model. We proposed Clique Averaging to remove the bias in gradient computation due to non-homogeneous averaging neighborhood by averaging gradients only with other nodes within the clique. Clique Averaging can in turn be used to implement an effective momentum. Through our extensive set of experiments, we showed that the clique structure of D-Cliques is critical in obtaining these results and that a small-world inter-clique topology with only $O(n \log n)$ edges achieves a very good compromise between convergence speed and scalability with the number of nodes.

D-Cliques thus appears to be very promising to reduce bandwidth usage on FL servers and to implement fully decentralized alternatives in a wider range of applications where global coordination is impossible or costly. For instance, the relative frequency of classes in each node could be computed using PushSum (Kempe et al., 2003), and the topology could be constructed in a decentralized and adaptive way with PeerSampling (Jelasity et al., 2007). This will be investigated in future work. We also believe that our ideas can be useful to deal with more general types of data heterogeneity beyond the important case of label distribution skew on which we focused in this paper. An important example is covariate shift or feature distribution skew (Kairouz et al., 2021), for which local density estimates could be used as basis to construct cliques that approximately recover the global distribution.

---

[4]We also observed that (Tang et al., 2018) is subject to numerical instabilities when run on topologies other than rings. When the rows and columns of W do not exactly sum to 1 (due to finite precision), these small differences get amplified by the proposed updates and make the algorithm diverge.

REFERENCES

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions. In *ICML*, 2016.

Dinh, C. T., Tran, N. H., and Nguyen, T. D. Personalized Federated Learning with Moreau Envelopes. In *NeurIPS*, 2020.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.

Esfandiari, Y., Tan, S. Y., Jiang, Z., Balu, A., Herron, E., Hegde, C., and Sarkar, S. Cross-Gradient Aggregation for Decentralized Learning from Non-IID data. Technical report, arXiv:2103.02051, 2021.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *NeurIPS*, 2020.

Hanzely, F., Hanzely, S., Horváth, S., and Richtarik, P. Lower Bounds and Optimal Algorithms for Personalized Federated Learning. In *NeurIPS*, 2020.

Hegedüs, I., Danner, G., and Jelasity, M. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. The Non-IID Data Quagmire of Decentralized Machine Learning. In *ICML*, 2020.

Jelasity, M., Montresor, A., and Babaoglu, Ö. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005. doi: 10.1145/1082469.1082470. URL https://doi.org/10.1145/1082469.1082470.

Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and Van Steen, M. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3): 8–es, 2007.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A.,

Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: Stochastic Controlled Averaging for On-Device Federated Learning. In *ICML*, 2020.

Kempe, D., Dobra, A., and Gehrke, J. Gossip-based Computation of Aggregate Information. *Foundations of Computer Science*, 2003.

Kong, L., Lin, T., Koloskova, A., Jaggi, M., and Stich, S. U. Consensus Control for Decentralized Deep Learning. Technical report, arXiv:2102.04828, 2021.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Cortes, C., and Burges, C. J. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 2020.

Lee, J.-W., Oh, J., Lim, S., Yun, S.-Y., and Lee, J.-G. Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture. Technical report, arXiv:2012.03214, 2020.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated Optimization in Heterogeneous Networks. In *MLSys*, 2020.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS*, 2017.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous Decentralized Parallel Stochastic Gradient Descent. In *ICML*, 2018.

Lin, T., Karimireddy, S. P., Stich, S. U., and Jaggi, M. Quasi-Global Momentum: Accelerating Decentralized Deep Learning on Heterogeneous Data. Technical report, arXiv:2102.04761, 2021.

Marfoq, O., Xu, C., Neglia, G., and Vidal, R. Throughput-Optimal Topology Design for Cross-Silo Federated Learning. In *NeurIPS*, 2020.

Marfoq, O., Neglia, G., Bellet, A., Kameni, L., and Vidal, R. Federated Multi-Task Learning under a Mixture of Distributions. In *NeurIPS*, 2021.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Nedić, A., Olshevsky, A., and Rabbat, M. G. Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Neglia, G., Xu, C., Towsley, D., and Calbi, G. Decentralized gradient methods: does topology matter? In *AISTATS*, 2020.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated Multi-Task Learning. In *NIPS*, 2017.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on networking*, 11 (1):17–32, 2003.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D^2: Decentralized Training over Decentralized Data. In *ICML*, 2018.

Vanhaesebrouck, P., Bellet, A., and Tommasi, M. Decentralized Collaborative Learning of Personalized Models over Networks. In *AISTATS*, 2017.

Watts, D. J. *Small worlds: The dynamics of networks between order and randomness*. Princeton University Press, 2000.

Xiao, L. and Boyd, S. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

Zantedeschi, V., Bellet, A., and Tommasi, M. Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs. In *AISTATS*, 2020.

A DETAILS ON SMALL-WORLD INTER-CLIQUE TOPOLOGY

We present a more detailed and precise explanation of the algorithm to establish a small-world inter-clique topology (Algorithm 4). Algorithm 4 instantiates the function `inter` with a small-world inter-clique topology as described in Section 3.3. It adds a linear number of inter-clique edges by first arranging cliques on a ring. It then adds a logarithmic number of "finger" edges to other cliques on the ring chosen such that there is a constant number of edges added per set, on sets that are exponentially bigger the further away on the ring. "Finger" edges are added symmetrically on both sides of the ring to the cliques in each set that are closest to a given set. "Finger" edges are added for each clique on the ring, therefore adding in total a linear-logarithmic number of edges.

Algorithm 4 $smallworld(DC)$: adds $O(\#N \log(\#N))$ edges

1: **Require:** set of cliques DC (set of set of nodes)
2:    size of neighborhood ns (default 2)
3:    function $least\_edges(S, E)$ that returns one of the nodes in S with the least number of edges in E
4: $E \leftarrow \emptyset$ {Set of Edges}
5: $L \leftarrow [C \text{ for } C \in DC]$ {Arrange cliques in a list}
6: **for** $i \in \{1, \ldots, \#DC\}$ **do**
7:    **for** $offset \in \{2^x \text{ for } x \in \{0, \ldots, \lceil \log_2(\#DC) \rceil\}\}$ **do**
8:      **for** $k \in \{0, \ldots, ns - 1\}$ **do**
9:        $n \leftarrow least\_edges(L_i, E)$
10:        $m \leftarrow least\_edges(L_{(i+offset+k)\%\#DC}, E)$
11:        $E \leftarrow E \cup \{\{n, m\}\}$
12:        $n \leftarrow least\_edges(L_i, E)$
13:        $m \leftarrow least\_edges(L_{(i-offset-k)\%\#DC}, E)$
14:        $E \leftarrow E \cup \{\{n, m\}\}$
15: **return** E

Algorithm 4 expects a set of cliques DC, previously computed by Algorithm 2; a size of neighborhood ns, which is the number of finger edges to add per set of cliques, and a function $least\_edges$, which given a set of nodes S and an existing set of edges $E = \{\{i, j\}, \ldots\}$, returns one of the nodes in E with the least number of edges. It returns a new set of edges $\{\{i, j\}, \ldots\}$ with all edges added by the small-world topology.

The implementation first arranges the cliques of DC in a list, which represents the ring. Traversing the list with increasing indices is equivalent to traversing the ring in the clockwise direction, and inversely. Then, for every clique i on the ring from which we are computing the distance to others, a number of edges are added. All other cliques are implicitly arranged in mutually exclusive sets, with size

and at offset exponentially bigger (doubling at every step). Then for every of these sets, ns edges are added, both in the clockwise and counter-clockwise directions, always on the nodes with the least number of edges in each clique. The ring edges are implicitly added to the cliques at offset 1 in both directions.

B ADDITIONAL EXPERIMENTS ON SCALING BEHAVIOR WITH INCREASING NUMBER OF NODES

Section 4.5 compares the convergence speed of various inter-clique topologies at a scale of 1000 nodes. In this section, we show the effect of scaling the number of nodes, by comparing the convergence speed with 1, 10, 100, and 1000 nodes, and adjusting the batch size to maintain a constant number of updates per epoch. We present results for Ring, Fractal, Small-world, and Fully-Connected inter-clique topologies.

Figure 16 shows the results for MNIST. For all topologies, we notice a perfect scaling up to 100 nodes, i.e. the accuracy curves overlap, with low variance between nodes. Starting at 1000 nodes, there is a significant increase in variance between nodes and the convergence is slower, only marginally for Fully-Connected but significantly so for Fractal and Ring. Small-world has higher variance between nodes but maintains a convergence speed close to that of Fully-Connected.

Figure 17 shows the results for CIFAR10. When increasing from 1 to 10 nodes (resulting in a single fully-connected clique), there is actually a small increase both in final accuracy and convergence speed. We believe this increase is due to the gradient being computed with better representation of examples from all classes with 10 fully-connected non-IID nodes, while the gradient for a single non-IID node may have a slightly larger bias because the random sampling may allow more bias in the representation of classes in each batch. At a scale of 100 nodes, there is no difference between Fully-Connected and Fractal, as the connections are the same; however, a Ring already shows a significantly slower convergence. At 1000 nodes, the convergence significantly slows down for Fractal and Ring, while remaining close, albeit with a larger variance, to Fully-Connected. Similar to MNIST, Small-world has higher variance and slightly lower convergence speed than Fully-Connected but remains very close.

We therefore conclude that Fully-Connected and Small-world have good scaling properties in terms of convergence speed, and that the linear-logarithmic number of edges of Small-world makes it the best compromise between convergence speed and connectivity, and thus the best choice for efficient large-scale decentralized learning in practice.
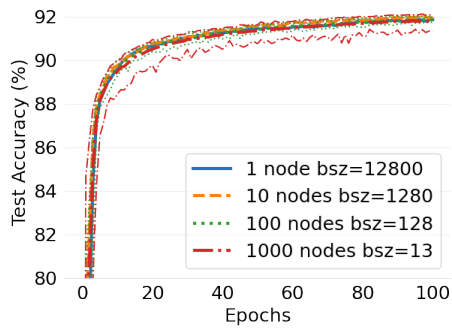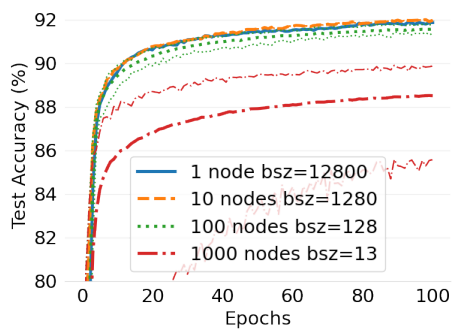
(a) Fully-Connected

(a) Fully-Connected
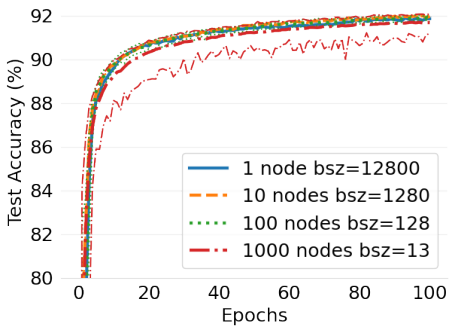
(b) Small-world

(b) Small-world
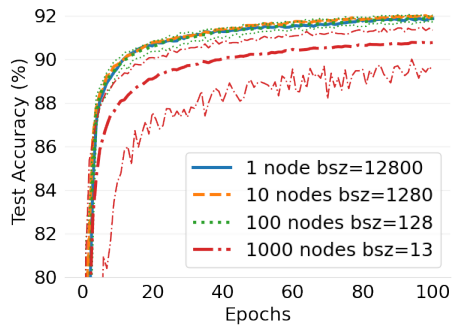
(c) Fractal

(c) Fractal

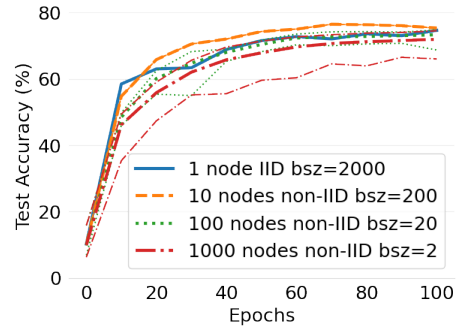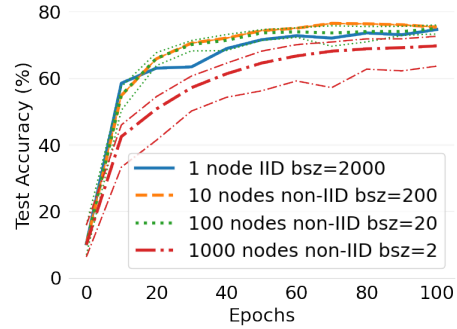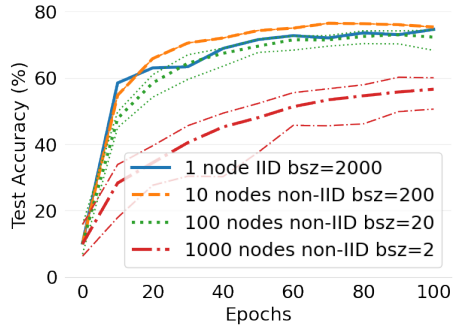(d) Ring

(d) Ring

Figure 16. MNIST: D-Cliques scaling behavior (constant updates per epoch and 10 nodes per clique) for different inter-clique topologies.
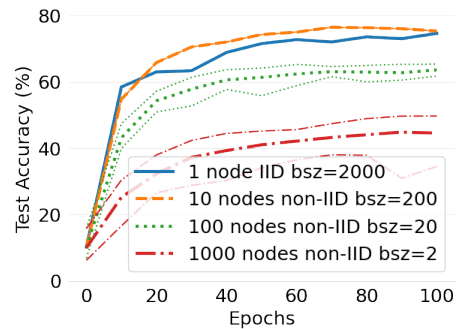
Figure 17. CIFAR10: D-Cliques scaling behavior (constant updates per epoch and 10 nodes per clique) for different inter-clique topologies.

C ADDITIONAL EXPERIMENTS WITH EXTREME LABEL SKEW

In this section, we present additional results for similar experiments as in Section 4 but in the presence of *extreme label distribution skew*: we consider that each node only has examples from a single class. This extreme partitioning case provides an upper bound on the effect of label distribution skew suggesting that D-Cliques should perform similarly or better in less extreme cases, as long as a small-enough average skew can be obtained on all cliques. In turn, this helps to provide insights on why D-Cliques work well, as well as to quantify the loss in convergence speed that may result from using construction algorithms that generate cliques with higher skew.

C.1 Data Heterogeneity Assumptions

To isolate the effect of label distribution skew from other potentially compounding factors, we make the following simplifying assumptions: (1) All classes are equally represented in the global dataset; (2) All classes are represented on the same number of nodes; (3) All nodes have the same number of examples.

While less realistic than the assumptions used Section 4, these assumptions are still reasonable because: (1) Global class imbalance equally affects the optimization process on a single node and is therefore not specific to the decentralized setting; (2) Our results do not exploit specific positions in the topology; (3) Imbalanced dataset sizes across nodes can be addressed for instance by appropriately weighting the individual loss functions.

These assumptions do make the construction of cliques slightly easier by making it easy to build cliques that have zero skew, as shown in Section C.2.

C.2 Constructing Ideal Cliques

Algorithm 5 shows the overall approach for constructing a D-Cliques topology under the assumptions of Section C.1.[5] It expects the following inputs: L, the set of all classes present in the global distribution $D = \bigcup_{i \in N} D_i$; N, the set of all nodes; a function $classes(S)$, which given a subset S of nodes in N returns the set of classes in their joint local distributions ($D_S = \bigcup_{i \in S} D_i$); a function $intra(DC)$, which given DC, a set of cliques (set of set of nodes), creates a set of edges ($\{\{i, j\}, \dots\}$) connecting all nodes within each clique to one another; a function $inter(DC)$, which given a set of cliques, creates a set of edges ($\{\{i, j\}, \dots\}$) connecting nodes belonging to differ-

---

[5]An IID version of D-Cliques, in which each node has an equal number of examples of all classes, can be implemented by picking $\#L$ nodes per clique at random.

ent cliques; and a function $weigths(E)$, which given a set of edges, returns the weighted matrix W_{ij}. Algorithm 5 returns both W_{ij}, for use in D-SGD (Algorithm 1 and 3), and DC, for use with Clique Averaging (Algorithm 3).

Algorithm 5 D-Cliques Construction

1: **Require:** set of classes globally present L,
2:   set of all nodes $N = \{1, 2, \dots, n\}$,
3:     fn $classes(S)$ that returns the classes present in a subset of nodes S,
4:     fn $intra(DC)$ that returns edges intraconnecting cliques of DC,
5:     fn $inter(DC)$ that returns edges interconnecting cliques of DC (Sec. 3.3)
6:   fn $weights(E)$ that assigns weights to edges in E
7: $R \leftarrow \{n \text{ for } n \in N\}$ {Remaining nodes}
8: $DC \leftarrow \emptyset$ {D-Cliques}
9: $C \leftarrow \emptyset$ {Current Clique}
10: **while** $R \neq \emptyset$ **do**
11:   $n \leftarrow$ pick 1 from $\{m \in R | classes(\{m\}) \subsetneq classes(C)\}$
12:   $R \leftarrow R \setminus \{n\}$
13:   $C \leftarrow C \cup \{n\}$
14:   **if** $classes(C) = L$ **then**
15:     $DC \leftarrow DC \cup \{C\}$
16:     $C \leftarrow \emptyset$
17: **return** $(weights(intra(DC) \cup inter(DC)), DC)$

The implementation builds a single clique by adding nodes with different classes until all classes of the global distribution are represented. Each clique is built sequentially until all nodes are parts of cliques. Because all classes are represented on an equal number of nodes, all cliques will have nodes of all classes. Furthermore, since nodes have examples of a single class, we are guaranteed a valid assignment is possible in a greedy manner. After cliques are created, edges are added and weights are assigned to edges, using the corresponding input functions.

C.3 Evaluation

In this section, we provide figures analogous to those of the main text using the partitioning scheme of Section C.1.

C.3.1 Data Heterogeneity is Significant at Multiple Levels of Node Skew

Figure 18 is consistent with Figure 1 albeit with slower convergence speed and higher variance. On the one hand, Figure 18 shows that an extreme skew amplifies the difficulty of learning. On the other hand, Figure 1 shows that the problem is not limited to the most extreme cases and is therefore worthy of consideration in designing decentralized federated learning solutions.

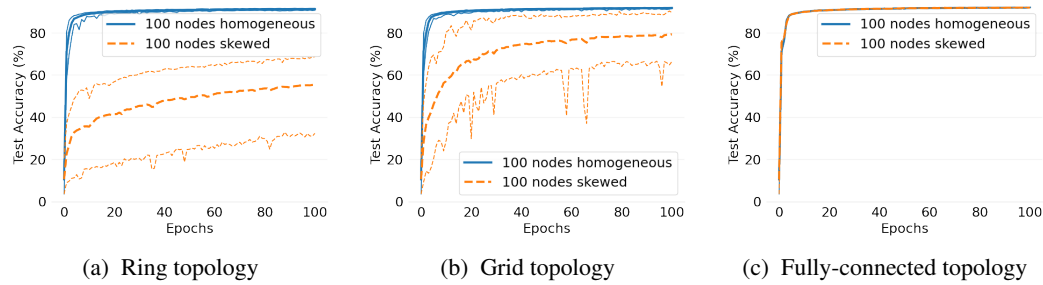(a) Ring topology   (b) Grid topology   (c) Fully-connected topology

Figure 18. Convergence speed of decentralized SGD with and without label distribution skew for different topologies on MNIST (Variation of Figure 1 using balanced classes and skewed with 1 class/node).

C.3.2 D-Cliques Match the Convergence Speed of Fully-Connected with a Fraction of the Edges

Figure 19 shows consistent results with Figure 5: D-Cliques work equally well in more extreme skew. It should therefore work well for other levels of label distribution skew commonly encountered in practice.

(a) MNIST   (b) CIFAR10 (with momentum)
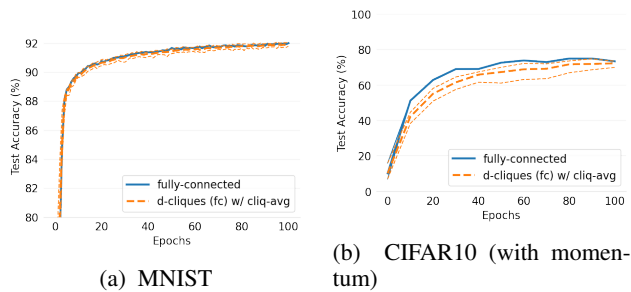
Figure 19. Comparison on 100 heterogeneous nodes between a fully-connected network and D-Cliques (fully-connected) constructed with Greedy Swap (10 cliques of 10 nodes) using Clique Averaging. (Variation of Figure 5 with 1 class/node instead of 2 shards/node).

C.3.3 Clique Averaging and Momentum are Beneficial and Sometimes Necessary

Figure 20 and Figure 21 show that, compared respectively to Figure 6 and Figure 7, Clique Averaging increases in importance the more extreme the skew is and provides consistent convergence speed at multiple levels.

C.3.4 D-Cliques Clustering is Necessary

In this experiment, we compare D-Cliques to different variations of random graphs, with additional variations compared to the experiments of Section 4.4, to show it is actually necessary. Compared to a random graph, D-Cliques enforce additional constraints and provide additional mechanisms: they ensure a diverse representation of all classes in the immediate neighbourhood of all nodes; they enable Clique Averaging to debias gradients; and they provide a
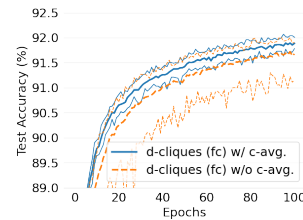

Figure 20. MNIST: Effect of Clique Averaging on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes). Y axis starts at 89. (Variation of Figure 6 with balanced classes and 1 class/node instead of 2 shards/node).
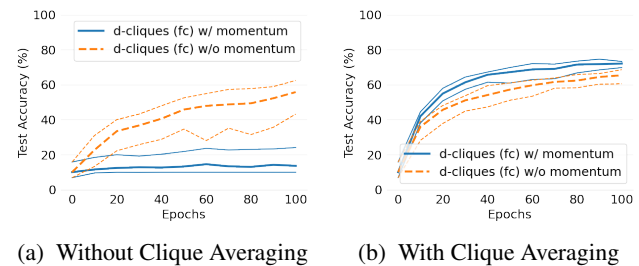

(a) Without Clique Averaging   (b) With Clique Averaging

Figure 21. CIFAR10: Effect of Clique Averaging, without and with momentum, on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes) (variation of Figure 7 with 1 class/node instead of 2 shards/node).

high-level of clustering, i.e. neighbors of a node are neighbors themselves, which tends to lower variance. In order to distinguish the effect of the first two from the last, we compare D-Cliques to other variations of random graphs: (1) with the additional constraint that all classes should be represented in the immediate neighborhood of all nodes (i.e. 'diverse neighbors'), and (2) in combination with unbiased gradients computed using the average of the gradients of a subset of neighbors of a node such that the skew of that subset is 0.

The partitioning scheme we use (Section C.1) makes the construction of both D-Cliques and diverse random graphs easy and ensures that in both cases the skew of the cliques or neighborhood subset is exactly 0. This removes the challenge of designing topology optimization algorithms for both D-Cliques and random graphs that would guarantee reaching the same level of skews in both cases to make results comparable.

(a) Linear   (b) Super- and Quasi-Linear

Figure 23. MNIST: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) with different inter-clique topologies. (variation of Figure 9 with 1 class/node instead of 2 shards/node).
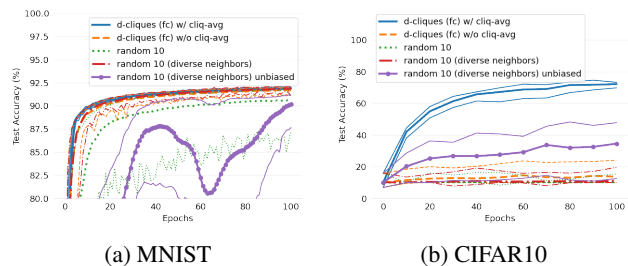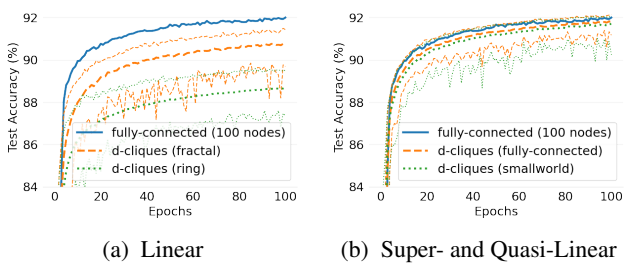

(a) MNIST   (b) CIFAR10

Figure 22. Comparison to variations of Random Graph with 10 edges per node on 100 nodes (variation of Figure 8 with 1 class/node instead of 2 shards/node as well as additional random graphs with more constraints).

Figure 22 compares the convergence speed of D-Cliques with all the variations of random graphs on both MNIST and CIFAR10. In both cases, D-Cliques converge faster than all other options. In addition, in the case of CIFAR10, the clustering appears to be critical for good convergence speed: even a random graph with diverse neighborhoods and unbiased gradients converges significantly slower.

C.3.5 D-Cliques Scale with Sparser Inter-Clique Topologies

Figure 23 and Figure 24 are consistent with Figure 9 and Figure 10. The less extreme skew enables a slightly faster convergence rate in the case of CIFAR10 (Figure 10).

C.3.6 Full Intra-Clique Connectivity is Necessary

Figure 25 and Figure 26 show higher variance than Figure 11 and Figure 12, with a significantly lower convergence speed in the case of CIFAR10 (Figure 26).

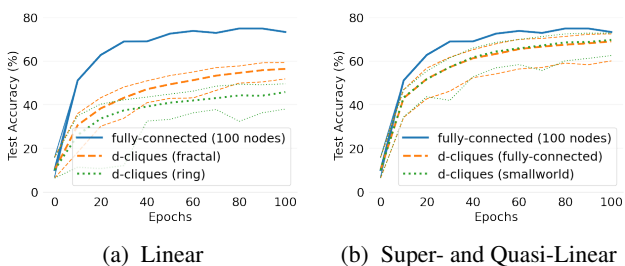(a) Linear   (b) Super- and Quasi-Linear

Figure 24. CIFAR10: D-Cliques convergence speed with 1000 nodes (10 nodes per clique, same number of updates per epoch as 100 nodes, i.e. batch-size 10x less per node) with different inter-clique topologies (variation of Figure 10 with 1 class/node instead of 2 shards/node).

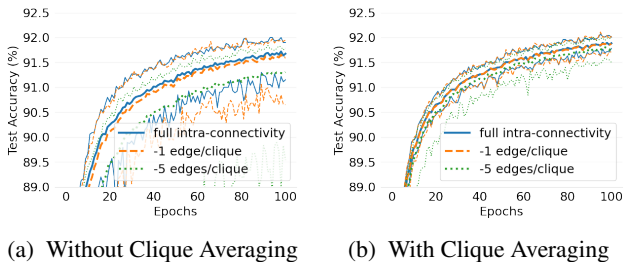(a) Without Clique Averaging   (b) With Clique Averaging

Figure 25. MNIST: Impact of intra-clique edge removal on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes) (variation of Figure 11 with 1 class/node instead of 2 shards/node). Y axis starts at 89.
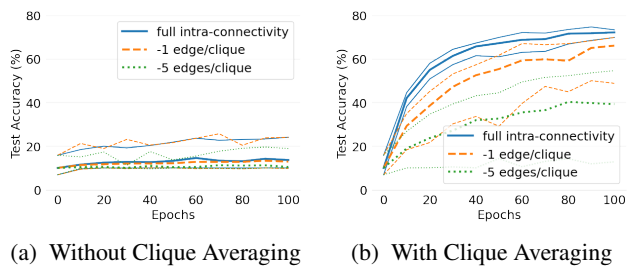
(a) Without Clique Averaging   (b) With Clique Averaging

Figure 26. CIFAR10: Impact of intra-clique edge removal (with momentum) on D-Cliques (fully-connected) with 10 cliques of 10 heterogeneous nodes (100 nodes) (variation of Figure 12 with 1 class/node instead of 2 shards/node).